1 Problem 1

Develop a program that transforms the image `img_lena.tif`, by the Haar wavelet.

(a) Original Image

(b) Haar Transformed Image

Figure 1

I performed a discrete Haar wavelet transformation to the image of Lena provided in `img_lena.tif` using the algorithm presented in class, which used discrete sums and differences of neighboring pixels. The output takes on a reconstructed quadtree form, and shows two levels of wavelet decomposition.

`problem1.py`

```
# Athanasios Athanassiadis February 2012
from mywavelet import *
from scipy.misc import imread
from pylab import imsave, cm

im  = imread('img_lena.tif')

n = 2
im_haar = haar_decompose(im, n)

imsave('my_lena_haar_{}.png'.format(n), im_haar, cmap=cm.gray)
```

2 Problem 2

Why do the detail images in the wavelet transform enhance horizontal, vertical, and diagonal edges?

To understand the detail images it is helpful to consider creating them by consecutively performing two 1D discrete wavelet transforms in each principal direction in the image. The output of the each pass of the 1D DWT returns two components of the input image, `cL` and `cH` which are the low and high frequency components of the image with respect to the decomposition wavelet used. In the case of the Haar wavelet, these components are simply the normalized sums and differences of neighboring pixels.

If the 1D DWT is run across horizontal pixels, then horizontal borders are in the direction of the pass, and therefore comprise a part of the `cL1` signal since the difference of neighboring pixels along a horizontal border is small. Because vertical and diagonal borders are not in the same direction as the pass, the differences of neighboring pixels along the pass will be high at a vertical or diagonal border, and these borders will be contained in the `cH1` output of the 1D DWT.

In the second pass, both the `cL1` and `cH1` output of the first pass are run through the 1D DWT algorithm again, but in the vertical direction. When the `cL1` image is run through a vertical DWT, horizontal borders will now be perpendicular to the pass, and will appear as a part of the `cH2` output of this run. The low frequency output `cL2` will then just be a smoothed version of the original image. When the `cH1` image is run through the vertical DWT, vertical borders will be along the pass, and will therefore constitute low frequency signals, output in `cL3`. Diagonal borders, on the other hand, will stil not be in the direction of the pass, and will be contained in the high frequency output `cH3`.

In this way, the output of the DWT consists of four images, `cL2`, `cH2`, `cL3`, `cH3`. Furthermore the detail images (all but `cL2`) represent enhanced horizontal, vertical, and diagonal boundaries of the original image.

3 Appendix: Common Code

The base code used for Problem 1 in this homework is contained in `mywavelet.py`.

mywavelet.py

```
# Athanasios Athanassiadis February 2012
import numpy as np
import pylab as pl

def haar_decompose(im_orig,n=2):
    '''
    haar_decompose(im,n)
        Haar wavelet decomposition of an image - this can be performed
        discretely by just calculating the sums and differences between
        neighboring samples, and passing the results recursively

    INPUTS:
        im  :   source image
        n   :   number of iterations (req: 2**n <= min(im.shape))

    '''
    im = im_orig.copy().astype(np.int64)
    if n==0:
        return im_orig

    shape = np.array(im.shape)
    im_haar = np.zeros(2 * (shape / 2))

    # cap the max of n
    n = min(n,np.log2(min(shape)))

    # initialize first step images (for after horizontal transform)
    imLF = np.zeros((shape[0],shape[1]/2))
    imHF = np.zeros((shape[0],shape[1]/2))

    # initialize second step images (final quadrants)
    cA = np.zeros(shape/2)  # main image DC components
    cV = np.zeros(shape/2)  # vertical edge-enhanced
    cH = np.zeros(shape/2)  # horizontal edge-enhanced
    cD = np.zeros(shape/2)  # diagonal edge-enhanced

    # calculate 1D DWT horizontally ( & account for odd number of rows)
```

```
    for i in range(shape[0] - shape[0]%2):
        cL1,cH1 = harr1D(im[i,:])
        imLF[i,:] = cL1
        imHF[i,:] = cH1

    # calculate 1D DWT vertically
    for j in range(shape[1]/2):
        cL2,cH2 = harr1D(imLF[:,j])
        cL3,cH3 = harr1D(imHF[:,j])

        # low freq of low freq is cA
        # high freq of low freq is cH
        # low freq of high freq is cV
        # high freq of high freq is cD
        cA[:,j] = cL2
        cH[:,j] = cH2
        cV[:,j] = cL3
        cD[:,j] = cH3

    # fill harr image from top right corner, leaving recursion for the end
    im_haar[:shape[0]/2,shape[1]/2:] = cV
    im_haar[shape[0]/2:,shape[1]/2:] = cD
    im_haar[shape[0]/2:,:shape[1]/2] = cH
    im_haar[:shape[0]/2,:shape[1]/2] = haar_decompose(cA, n-1)

    return im_haar

def harr1D(sig):
    '''
    harr1d(sig)
        takes a 1D signal and returns its discrete Harr wavelet transform
        in the form of the high and low frequency components of the signal

    INPUTS
        sig :   input signal

    '''
    cL = np.zeros(len(sig)/2)
    cH = np.zeros(len(sig)/2)

    # takes sums/differences for HF and LF components, and normalize
    for i in range(len(sig)/2-1):
        cL[i] = 1.0 * (sig[2*i+1] + sig[2*i]) / np.sqrt(2)
        cH[i] = 1.0 * (sig[2*i+1] - sig[2*i]) / np.sqrt(2)


    return cL,cH
```