1.

Haar transform for each 8x8 block can be computed by:

\[
c = (H_{R_8}^T)f(H_{R_8})
\]

where \(f\) is an 8x8 image block, \(c\) is the 8x8 transform coefficients and \(H_{R_8}\) is the Haar transform matrix as given in slide no.16 of the “Unitary Transform” lecture notes.

DCT can be carried out using the MATLAB function ‘dct2( )’.

(a) The average energy of the image set is found by squaring each pixel and then averaging over the entire image set (mean squared value of the image pixels).

The average energy of the transform coefficients is found by squaring each coefficient and then averaging over all the coefficients (mean squared value of the transform coefficients).

The average energy of the image set, the HT coefficients, and the DCT coefficients are the same, which is 12307.51. This verifies the energy conservation property of HT and DCT.

(b) The following matrices show the descending order of the HT and DCT coefficients computed using our image set.

### Haar Coefficients

\[
\begin{array}{cccccccc}
1 & 2 & 5 & 4 & 16 & 19 & 17 & 10 \\
3 & 7 & 12 & 13 & 32 & 31 & 29 & 30 \\
6 & 14 & 20 & 21 & 39 & 44 & 48 & 43 \\
8 & 15 & 23 & 18 & 42 & 47 & 45 & 46 \\
9 & 25 & 34 & 37 & 54 & 56 & 63 & 50 \\
22 & 28 & 36 & 33 & 55 & 59 & 53 & 52 \\
24 & 27 & 40 & 35 & 58 & 49 & 61 & 60 \\
11 & 26 & 41 & 38 & 51 & 64 & 60 & 64
\end{array}
\]

### DCT Coefficients

\[
\begin{array}{cccccccc}
1 & 2 & 4 & 8 & 14 & 19 & 28 & 46 \\
3 & 6 & 7 & 15 & 21 & 31 & 44 & 51 \\
5 & 10 & 12 & 17 & 25 & 35 & 43 & 53 \\
9 & 13 & 18 & 22 & 30 & 38 & 49 & 57 \\
11 & 20 & 23 & 27 & 34 & 45 & 52 & 58 \\
16 & 26 & 29 & 36 & 42 & 50 & 56 & 61 \\
24 & 33 & 37 & 40 & 47 & 55 & 60 & 63 \\
32 & 39 & 41 & 48 & 54 & 59 & 62 & 64
\end{array}
\]
(c) The figure below shows that DCT has better energy concentration than HT.
(a) The mean squared values of the KLT coefficients $c_i$ are:

$$E[c_i^2] = [R_{xx}]_{i,j} = \Lambda_{i,j}$$

where $\Lambda$ is the diagonal matrix with eigenvalues of the autocorrelation matrix $R_{xx}$ on its diagonal (slides no.5~6 of the “Unitary Transform” lecture notes).

This can be done by first measuring the autocorrelation matrix averaged overall all 8x8 blocks. The eigenvalues of the autocorrelation matrix can be found by the eig( ) function in Matlab.

(b) KLT has energy concentration property since it is a unitary transform. It can be seen also from the fact that the mean squared value of the KLT coefficients is the same as the mean squared value of the image set.

The figure below shows that KLT has the best energy concentration property, but it is only slightly better than DCT.
clear; close all;

% read the test set into a 256x256x3 matrix
m = 256; n = 256;
img = zeros(m,n,3);
img(:,:,1) = imread('mri.tif');
img(:,:,2) = imread('einstein.tif');
img(:,:,3) = imread('smandril.tif');

%% probelm 1

% squared value averaged over pixels, i.e. pixel domain average energy
ms_img = mean(img(:).^2);

% there are (m/8)x(n/8) of 8x8 tranform coefficient blocks for each image
dct = zeros(8,8,m/8,n/8,3);
haar = zeros(8,8,m/8,n/8,3);

% taking transforms for each block
for r = 1:m/8
    for c = 1:n/8
        for i = 1:3
            haar(:,:,r,c,i) = haar8by8(img((8*(r-1)+1):(8*r),(8*(c-1)+1):(8*c),i));
            dct(:,:,r,c,i) = dct2(img((8*(r-1)+1):(8*r),(8*(c-1)+1):(8*c),i));
        end
    end
end

% squared value averaged over blocks, i.e. mean squared value for each coefficient
msb_haar = mean(mean(mean(haar.^2,5),4),3);
msb_dct = mean(mean(mean(dct.^2,5),4),3);

% squared value averaged over coefficients, i.e. transform domain average energy
ms_haar = mean(msb_haar(:));
ms_dct = mean(msb_dct(:));

disp(sprintf('pixel domain average energy = %f',ms_img));
disp(sprintf('transform domain average energy (Haar) = %f',ms_haar));
disp(sprintf('transform domain average energy (DCT) = %f',ms_dct));

% sort the mean squared value for coefficients (descending)
% first sort into ascending order
[msb_haar_ascending,idx_haar] = sort(msb_haar(:));
[msb_dct_ascending,idx_dct] = sort(msb_dct(:));
% convert to descending order
msb_haar_descending = flipud(msb_haar_ascending);
msb_dct_descending = flipud(msb_dct_ascending);
% idx is the index indicating the order of the coefficients
[dump,idx_haar] = sort(flipud(idx_haar));
dump,idx_dct] = sort(flipud(idx_dct));
% reshape the column vector into a 8x8 matrix
idx_haar = reshape(idx_haar,8,8);
idx_dct = reshape(idx_dct,8,8);
display(idx_haar);
display(idx_dct);

% compute accumulated ratio
cum_ratio_haar = cumsum(msb_haar_descending)/sum(msb_haar_descending)*100;
cum_ratio_dct = cumsum(msb_dct_descending)/sum(msb_dct_descending)*100;

% plot graph
k = 1:8^2;
figure; plot(k,cum_ratio_haar,'-',k,cum_ratio_dct,'-.');
legend('HT','DCT');
title('Percentage of total energy contained in the first k coefficients');
xlabel('number of coefficients, k');
ylabel('Percentage of total energy (%)');
%% problem 2

% auto-correlation for each block
Rtemp = zeros(64,64,m/8,n/8,3);
for r = 1:m/8
  for c = 1:n/8
    for i = 1:3
      block = img((8*(r-1)+1):(8*r),(8*(c-1)+1):(8*c),i);
      Rtemp(:,:,r,c,i) = block(:)*block(:)';
    end
  end
end

% expected value of auto-correlation
Rff = mean(mean(mean(Rtemp,5),4),3);

% eig(Rff) is eigenvalues of Rff, i.e. diagonal of auto-correlation of the KLT coefficients
% i.e. mean squared value of each KLT coefficient
msb_klt = eig(Rff);

% squared value averaged over coefficients, i.e. transform domain average energy
ms_klt = mean(msb_klt);
disp(sprintf('transform domain average energy (KLT) = %f',ms_klt));

% sort into descending order and compute accumulated ratio
msb_klt_descending = flipud(sort(msb_klt));
cum_ratio_klt = cumsum(msb_klt_descending)/sum(msb_klt_descending)*100;

% plot graph
figure; plot(k,cum_ratio_haar,'-',k,cum_ratio_dct,'-.',k,cum_ratio_klt,:);
legend('HT','DCT','KLT');
title('Percentage of total energy contained in the first k coefficients');
xlabel('number of coefficients, k');
ylabel('percentage of total energy (%)');

function Y = haar8by8(X)
% Haar transform for an 8x8 block

s = sqrt(2);
Hr8 = [ 1 1 s 0 2 0 0 0;...
    1 1 s 0 -2 0 0 0;...
    1 -s 0 0 2 0 0 0;...
    1 1 -s 0 0 -2 0 0;...
    1 -1 0 s 0 0 2 0;...
    1 -1 0 s 0 0 -2 0;...
    1 -1 -s 0 0 0 2;...
    1 -1 0 -s 0 0 0 -2] / sqrt(8);
Y = Hr8' * X * Hr8;