Assignment 01: Introduction to MatLab and Octave

1 Warming up with images:

- Octave Script: (`script1.m`)

```
clear

A(1,26:50) = ([1:25]./25).^2;          % A row vector of 50 elements
X(1,1:50) = A;                         % put A at the begining of X
X(1,51:100) = fliplr(-A);              % Flipping A and Put into X
Y = uint8((((ones(50,1) * X)+1)./2)*255); % 50 duplicates row and ...
    scaling the values into the range 0 to 255

colormap(gray(256));
image(Y+1);
print -dpng cornsweet.png
```

- Output Image: (`cornsweet.png`)

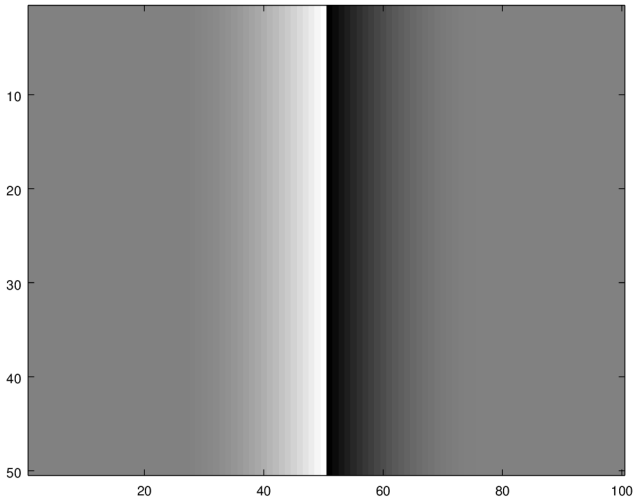

Figure 1: Resulted image having Cornsweet effect.

2 The Koch Curve:

- Octave function for Koch generation: (`koch.m`)

```
function Kn_1 = koch(Kn)
    % Kn_1 = koch(Kn)
    % koch function that calculates Kn+1 from Kn. Kn contains all ...
        points as arguments.
    % As a return we will get all points of Kn+1 in Kn_1 matrix.

    co = size(Kn, 1);          % Number of points in Kn
    count = co .+ 3.*(co-1);  % Number of points in Kn_1
    Kn_1 = zeros(count,2);

    % In the following, I have generate points from Kn and place ...
        them in the right order
    Kn_1(1:4:count,1) = Kn(1:co,1); % placing all X0
    Kn_1(1:4:count,2) = Kn(1:co,2);

    % generate and place all X2 in Kn_1
    Kn_1(2:4:count,1) = .667 .* Kn(1:co-1,1) + .333 .* Kn(2:co,1);
    Kn_1(2:4:count,2) = .667 .* Kn(1:co-1,2) + .333 .* Kn(2:co,2);

    % generate and place all X3 in Kn_1
    sub_eq1 = .288 .* (Kn(1:co-1,2) - Kn(2:co,2));
    sub_eq2 = .288 .* (Kn(2:co,1) - Kn(1:co-1,1));
    Kn_1(3:4:count,1) = .5 .* Kn(1:co-1,1) + .5 .* Kn(2:co,1) + ...
        sub_eq1;
    Kn_1(3:4:count,2) = .5 .* Kn(1:co-1,2) + .5 .* Kn(2:co,2) + ...
        sub_eq2;

    % generate and place all X4 in Kn_1
    Kn_1(4:4:count,1) = .333 .* Kn(1:co-1,1) + .667 .* Kn(2:co,1);
    Kn_1(4:4:count,2) = .333 .* Kn(1:co-1,2) + .667 .* Kn(2:co,2);

endfunction
```

- Octave Script as main function: (`script2.m`)

```
clear
tic                          %start time count

K0 = [0 0; 1 0];             %initial value
K1 = koch(K0);
K2 = koch(K1);
K3 = koch(K2);
K4 = koch(K3);
K5 = koch(K4);
K6 = koch(K5);
K7 = koch(K6);
K8 = koch(K7);
K9 = koch(K8);
K10 = koch(K9);

toc                          %Give Elapsed time
plot(K10(:,1),K10(:,2));
print -dpng foo.png
```

- Output Image: (`foo.png`)

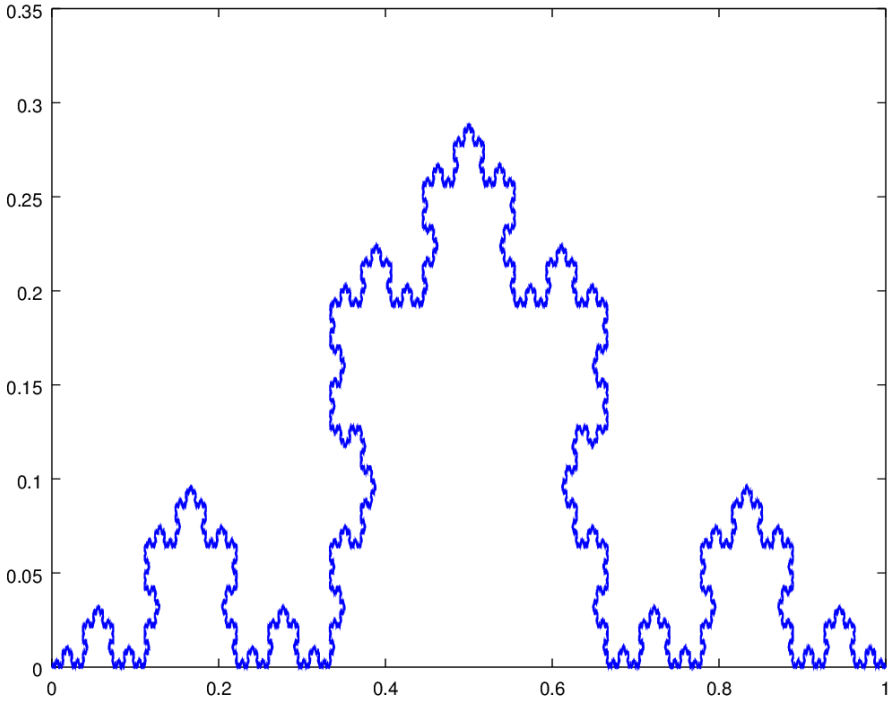

Figure 2: Plotting of koch curve for n = 10.

3 Haar Wavelets:

3.1 Part 01: Noisy Sine Curve:

- Octave Script as main function: (`script3.m`)

```
clear
x = 1:1:1024;
sin_noisy = sin(0:2*pi/1023:2*pi) + randn(1,1024)/10;
my_display(x, sin_noisy, 'Noisy Sine Curve', 'sine');

mat1 = decomposition_mat(1024); % Get Level-1 decomposition matrix

mat2 = decomposition_mat(512);  % Get Level-2 decomposition matrix
mat2 = append(mat2,1024);

mat3 = decomposition_mat(256);  % Get Level-3 decomposition matrix
mat3 = append(mat3,1024);

mat4 = decomposition_mat(128);  % Get Level-4 decomposition matrix
mat4 = append(mat4,1024);

mat5 = decomposition_mat(64);   % Get Level-5 decomposition matrix
mat5 = append(mat5,1024);

mat = mat1 * mat2 * mat3 * mat4 * mat5; % final decomposition matrix
y = sin_noisy * mat;
my_display(x, y, 'After applying all 5 decompositions', 'out');

cos_f = cos(0:2*pi/31:2*pi);  % generate cos curve
y = [cos_f, y(33:1024)];      % generate cos curve + high frequency
my_display(x,y, 'Cosine curve + High Frequency', 'cos');

rev = y * inv(mat);  % Recompostion done in a single operation
my_display(x,rev, 'Noisy Cosine', 'cos_noisy');
```

- Octave function that generate decomposition matrix: (`decomposition_mat.m`)

```
function X = decomposition_mat(n)
    % function X = decomposition_mat(n)
    % This function will generate a matrix X for decomposing an ...
        image (of size n x n) using Haar wavelet.
    % Here n should contain a value that is power of 2.

    full = n;
    half = n/2;
    mat1 = zeros(full,half);
    mat2 = zeros(full,half);

    temp = eye(half).* 0.5; % Here eye() is an builtin function to ...
        generate Identity matrix.
    mat1(1:2:full,:) = temp(1:half,:);
    mat1(2:2:full,:) = temp(1:half,:);
    mat2(1:2:full,:) = temp(1:half,:);
    mat2(2:2:full,:) = temp(1:half,:).*-1;
    X = [mat1, mat2];
endfunction
```

- Octave function that append additional area to the decomposition matrix: (`append.m`)

```
function x = append(mat, s)
    % function x = append(mat, s)
    % This function will padding the decomposition matrix 'mat' in ...
        such a way that
    % it will make the remaining(high frequency) part of the data ...
        unchanged.
    % x is the padded decomposition matrix and s is the desired ...
        dimention of x

    [r c] = size(mat);
    m = s-c;            % m is the additional lenth
    x1 = zeros(r,m);
    x2 = eye(m);        % eye() is a buitin function for generate ...
        Identity matrix
    x3 = zeros(m,r);
    x = [mat,x1;x3,x2];
endfunction
```

- Octave function that display plots: (`my_display.m`)

```
function my_display(x,y, msg, file_name)
    % This function will plot x and y having a msg (a level on the ...
        plot) on it.
    % Also print the plot as file_name.png

    plot(x,y);
    grid,axis([1 1200 -1.5 +1.5]);
    text (500, 1.2, msg);
    print(file_name,'-dpng');
endfunction
```

- Images of all plots:

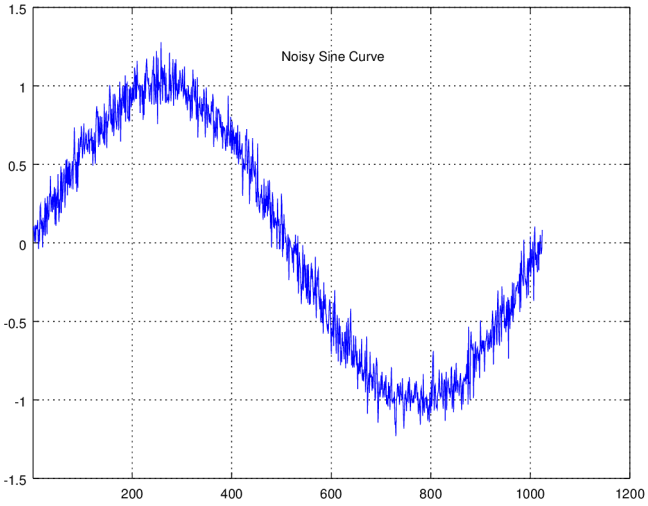

Figure 3: Noisy sine curve.

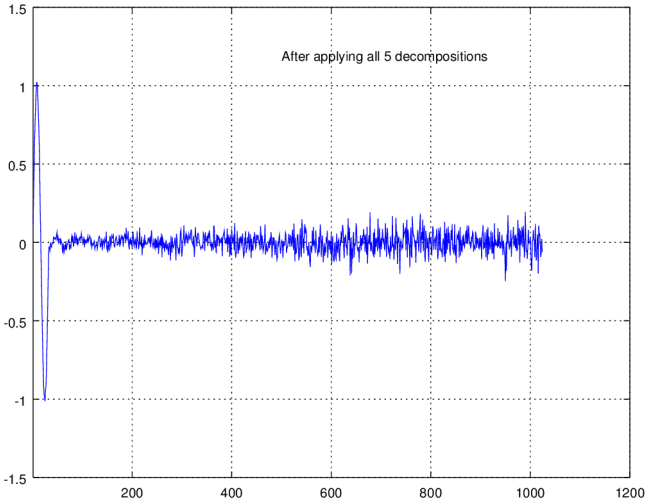

Figure 4: After applying all 5 Haar decompositions on noisy sine curve.

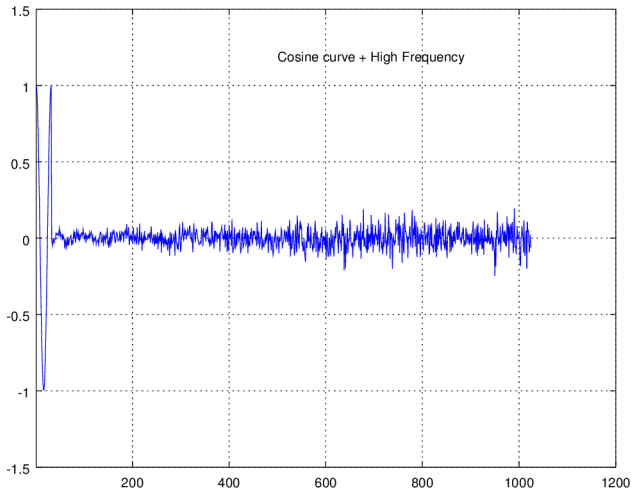

Figure 5: Cosine curve + high frequency portion.

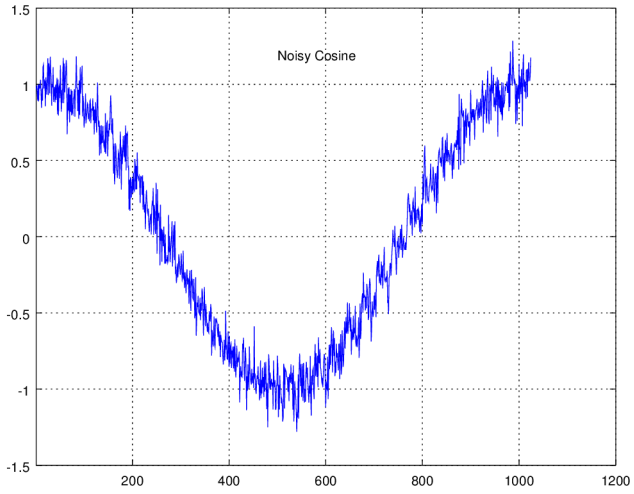

Figure 6: Noisy cosine curve after recomposition.

3.2 Part 02: Moving to images:

- Octave Script as main function: (`script4.m`)

```
clear
mat1 = decomposition_mat(1024); % docomposition matrix for level-1
mat2 = decomposition_mat(512);  % docomposition matrix for level-2
mat3 = decomposition_mat(256);  % docomposition matrix for level-3
x = double(imread('cat.png'));

% Applying Level-1 decomposition on whole input image.
y = ((x * mat1)' * mat1)';
% Applying Level-2 decomposition on low frequency image.
y(1:512,1:512) = ((y(1:512,1:512) * mat2)' * mat2)';
% Applying Level-3 decomposition on next low frequency image.
y(1:256,1:256) = ((y(1:256,1:256) * mat3)' * mat3)';

z = my_display(y); % create a well visible whole image
imwrite(z,'Haar_cat.png');
```

- Octave function for display: (`my_display.m`)

```
function y = my_display(y)
    % High frequency components contain low values.
    % Scaling only the high frequency components to the range 0 to 255.

    temp = y(1:128,1:128); % seperate the low frequency part from ...
        whole image.
    y(1:128,1:128) = 0;
    y = abs(y);
    y = y .- min(y(:));             %subtract the minimum value
    y = uint8((y./max(y(:)))*256);  % Scaling to 0 to 255

    y(1:128,1:128) = temp; % Appending the low frequency part.
endfunction
```

- Output Image: (`Haar_cat.png`)

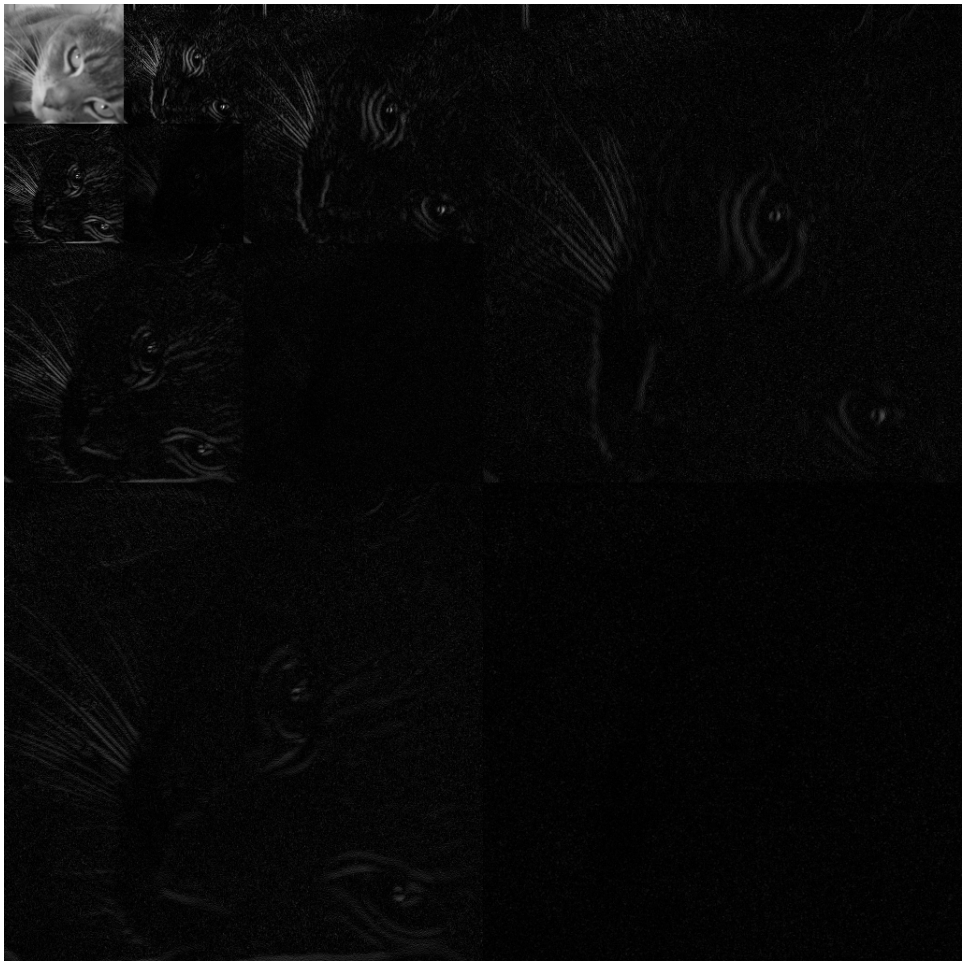

Figure 7: Resulted image after applying 3 level Haar wavelet decomposition.

- Questions and Answers:

  Question 1: *What does the removed high-frequency data look like? What is the range, min, max, and how does this affect how it looks when you draw it with image?*

  ANSWER: In Figure 8(a), the cat image after applying 3 level Haar decomposition is given. The black area in the image is the removed high frequency part. The surface plot of the same image is given in Figure 8(b).

  We can see in the surface plot that the high frequency part contains very low pixel values. The minimum and the maximum pixel values are -26.312 and 29.750 respectively in this part. But in the low frequency part (small cat image on the top left corner of the Figure 8(a)) the pixel range is from 15.562 to 233.09. So when I draw it (without rescaling), it shows a large black area and the details are very difficult to see in the image.

  Figure 7 is showing the corrected output image after applying value rescaling on the high frequency part. Now the details are visible with naked eyes.

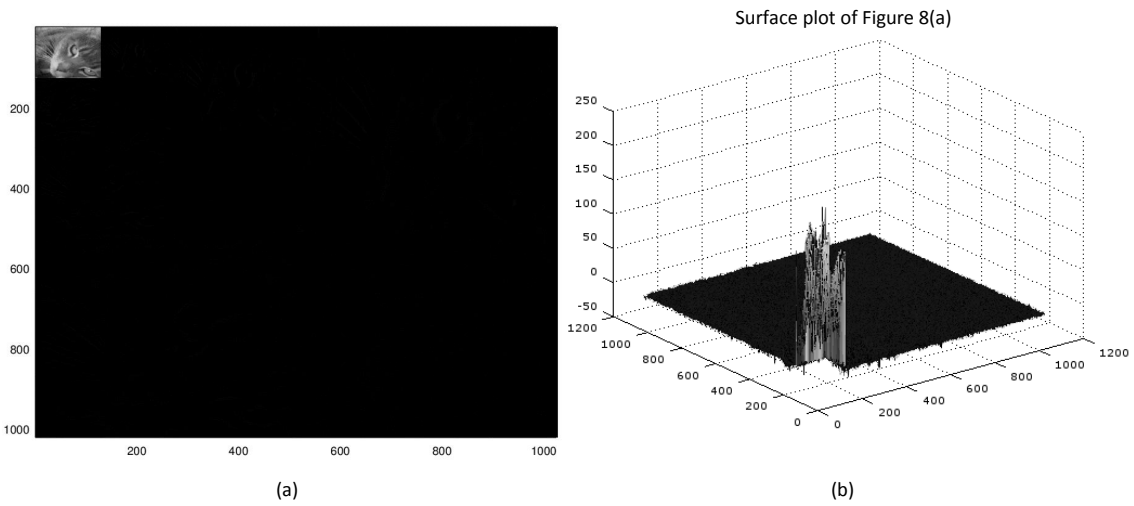

Figure 8: (a) After applying 3 level Haar decomposition on Cat image and (b) its surface plot.

  Question 2: *Discuss how Haar decomposition can be useful for image compression.*

  ANSWER: When we apply Haar decomposition on an image, it will create four frequency bands where one of them is the low frequency part (approximation of the original image) and left three are the high frequency part (horizontal, vertical and diagonal details of the image). These three high frequency part contain very low pixel values (close to zero) that contributes to the details (lines, points, edges) of the image.

  For compression, we can make most of the high frequency values to zero using simple thresholding and convert the image matrix into a sparse matrix. We know that a sparse matrix takes less space than a dense matrix.

After decompression, it will generate an image approximately close to the original image. Definitely, the decompressed image will slightly lose (or blur) the details (edge, lines, points) but in bare eyes its difficult to find the difference.

Question 3: *What are the limits of Haar decomposition? Why and when would you decompose to this limit?*

ANSWER: According to the equation of (ψ function) Haar decomposition, the number of data samples should be a power of 2 that means 2^n where n is a positive integer greater than 0. So we can apply maximum n successive Haar decomposition on a dataset of 2^n samples.

Number of level (limit) of Haar decomposition depends on how much high frequency data we need to handle for a specific purpose or application. When we apply successive Haar decomposition, it will reduce the approximation (low frequency data) of the data samples by 50% and increase the high frequency data by 50% at each iteration. If we need more high frequency data to separate, we can apply more level of Haar decomposition on the low frequency data until it reaches the limit of sample size. For example, if we apply n-level successive Haar decomposition to an image of 2^n by 2^n then we will get an image having high frequency data in all pixel positions except the top left corner pixel (it is a single pixel having the low frequency data). In image compression how many level of Haar decomposition we need is depends on the trade off between compression rate and decompressed image quality. So number of successive Haar decomposition depends on application.