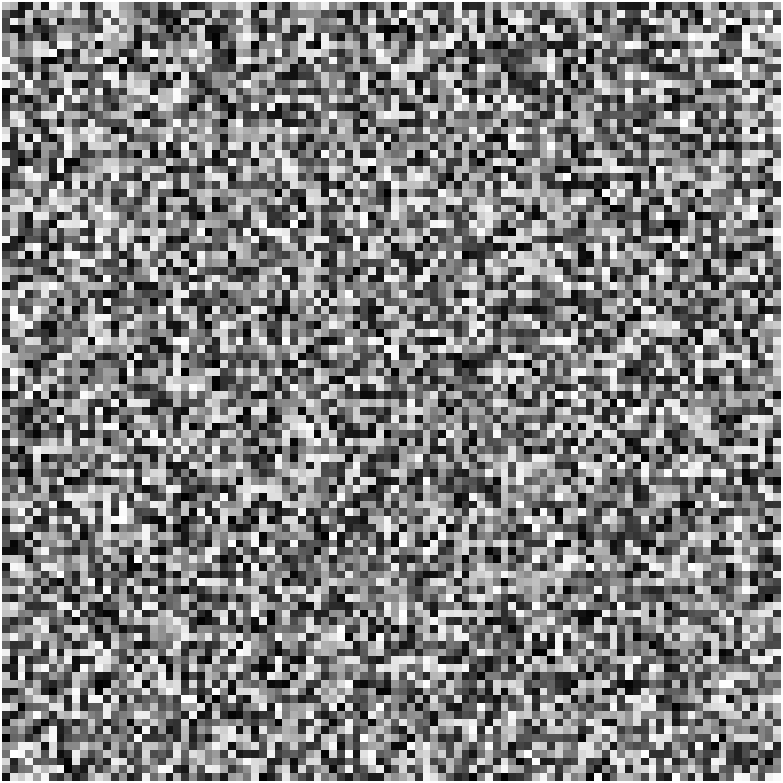
**Applied Computer Vision (CS-696)  
Assignment 1**

**Name – Dhaval Harish Sharma  
Red ID – 824654344**

**I. Using Matlab [60%]**

**The goal of this problem set is to become familiar with basic Matlab commands, practice manipulating vectors and matrices, and try out basic image display and plotting functions. If you are unsure what a Matlab function does, check the reference manual (at the command line, type ‘help’ and then the command name).**

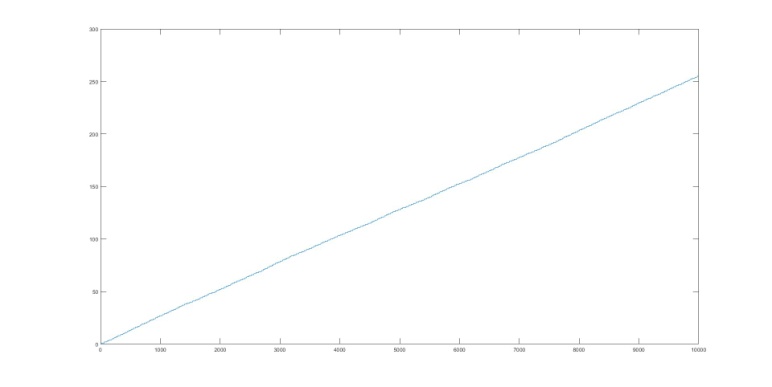
* 1. **Read over the provided Matlab introduction code and its comments in intro.m. Open an interactive session in Matlab and test the commands by typing them at the prompt. (You might skip this step if you are an expert on Matlab).**
  2. **Describe (in words where appropriate) the result of each of the following Matlab commands. Use the help command as needed, but try to determine the output without entering the commands into Matlab. Do not submit a screenshot of the result of typing these commands.**

1. **>> x = randperm(5);**  
   The full form of randperm is random permutation. It returns a single dimensional vector (size 1 \* n) of randomized n elements in range 1 to n without repeating it.  
   In this case, the number of elements is 5 which mean the output vector size would be 1 \* 5.  
   A sample output of this command would be [2, 1, 4, 5, 3].
2. **>> a = [1:10];  
   >> b = a([1:3:end]);**  
   [start:end] is an interval from start to end (both inclusive) with default step value of 1. We can also specify the step size explicitly by [start:step-size:end].  
   [1:10] creates a single dimensional vector with elements from 1 to 10. The resultant row vector is assigned to the variable a. So, a would be [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].  
   a([1:3:end]) traverses every third element of row vector a from first index to the last index. The elements traversed are stored in the variable b. So, b would be [1, 4, 7, 10].
3. **>>f = [1501:2000];  
   >>g = find(f > 1850);  
   >>h = f(g);**[1501:2000] creates a single dimensional vector with elements from 1501 to 2000. The resultant row vector is assigned to the variable f. So, f would be [1501, 1502, 1503, 1504, …, 1998, 1999, 2000].  
   Find command finds the indexes of given matrix which satisfies the condition mentioned in its arguments. So, here indexes of all the pixels of matrix f whose value is greater than 1850 are stored in matrix g.  
   Therefore, g would be [351, 352, 353, …, 498, 499, 500].  
   f(g) generates the values present in matrix f at indexes mentioned in the matrix g and stores them in matrix h.  
   Therefore, f(g) would be [1851, 1852, 1853, …, 1998, 1999, 2000].
4. **>> x = 22.\*ones(1, 10);  
   >> y = sum(x);**  
   ones(m, n) creates a matrix of size m \* n with all elements as 1. “.\*” represents element wise multiplication. So, all 1 of matrix with size m \* n are multiplied by the constant 22.  
   So, by executing above line, x would be a matrix of size 1\*10 with all elements 22.  
   The vector x would be [22, 22, 22, …, 22, 22] with 10 elements.  
   sum(matrix) returns matrix with size 1 \* n if there are more than one rows; where n is the number of columns. Each column contains value equals to sum of corresponding column in the original matrix.  
   In case of 1 row, sum(matrix) performs the sum of all elements in the matrix. In our case, y would be 220.
5. **>> a = [1:1000];  
   >> b = a([end: -1: 1]);**  
   [1:1000] creates a single dimensional vector with elements from 1 to 1000. And row vector is assigned to the variable a. So, a would be [1, 2, 3, 4, 5, …, 997, 998, 999, 1000].  
   In case of b, the traversal starts from end element of matrix a and ends at the first element. Note that step value is -1 which means we are decreasing index while traversing. This results in a reversed list i.e. the matrix b will be the reverse of matrix a. So, b would be [1000, 999, 998, …, 3, 2, 1].
   1. **Given a 100 x 100 uint8 matrix A representing a grayscale image, write a few lines of code to do each of the following. Try to avoid using loops.  
      (a) Sort all the intensities in A, put the result in a single 10,000-dimensional vector x, and plot the values in x.  
      (b) Display a figure showing a histogram of A’s intensities with 32 bins.  
      (c) Create and display a new binary image the same size as A, which is white wherever the intensity in A is greater than a threshold t, and black everywhere else.  
      (d) Generate a new image (matrix), which is the same as A, but with A’s mean intensity value subtracted from each pixel. Set any negative values to 0.  
      (e) Use rand to write a function that returns the roll of a six-sided die.  
      (f) Let y be the vector: y = [1:6]. Use the reshape command to form a new matrix z whose first column is [1, 2, 3]’, and whose second column is [4, 5, 6]’.  
      (g) Use the min and find functions to set x to the single minimum value that occurs in A, and set r to the row it occurs in and c to the column it occurs in.  
      (h) Let v be the vector: v = [1 8 8 2 1 3 9 8]. Use the unique function to compute the total number of unique values that occur in v.  
      Solution:**Random image is generated by the following code,  
      arr = rand(100, 100) \* 255;  
      A = uint8(arr);  
      imshow(A);  
        
      Output:  
      ****  
       **(a) Sort all the intensities in A, put the result in a single 10,000-dimensional vector x,and plot the values in x.**Code:  
      [row, col] = size(A);

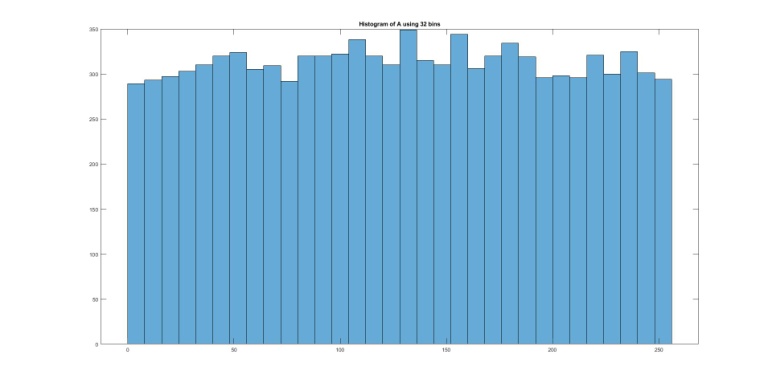
length = row \* col;

x = reshape(A, 1, length);

x = sort(x);

plot(x);  
  
Output:  
  
**(b) Display a figure showing a histogram of A’s intensities with 32 bins.**Code:  
figure;

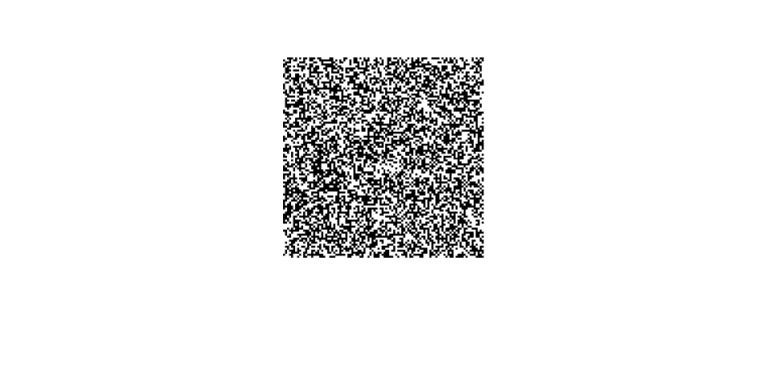
histogram(A, 32);

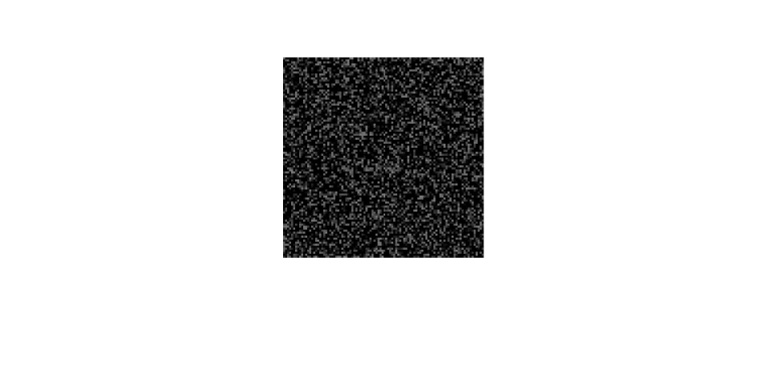
title("Histogram of A using 32 bins");  
  
Output:  


**(c) Create and display a new binary image the same size as A, which is white wherever the intensity in A is greater than a threshold t, and black everywhere else.**Code:  
figure;

b = img > t;

imshow(b);  
  
Output:

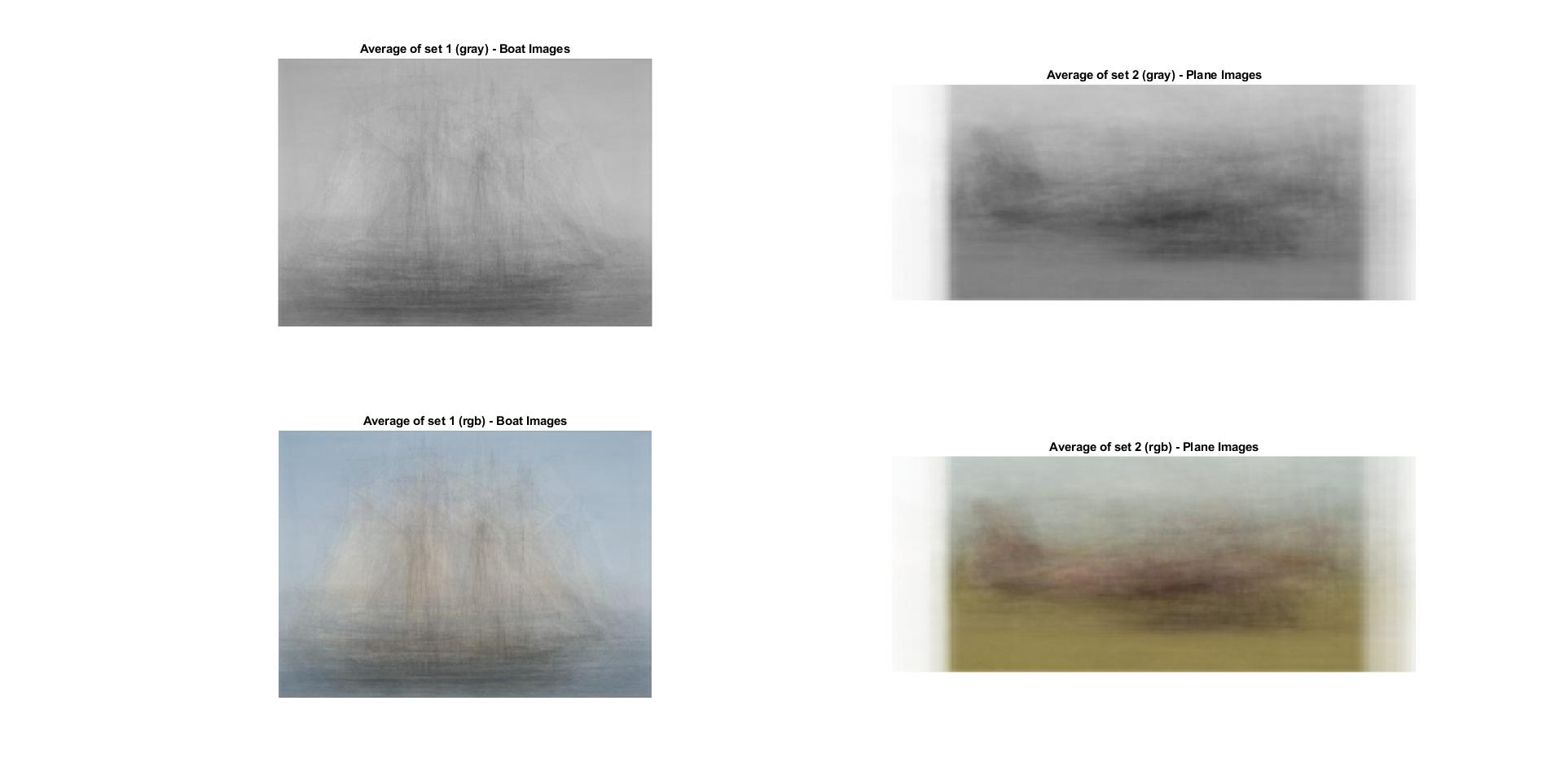
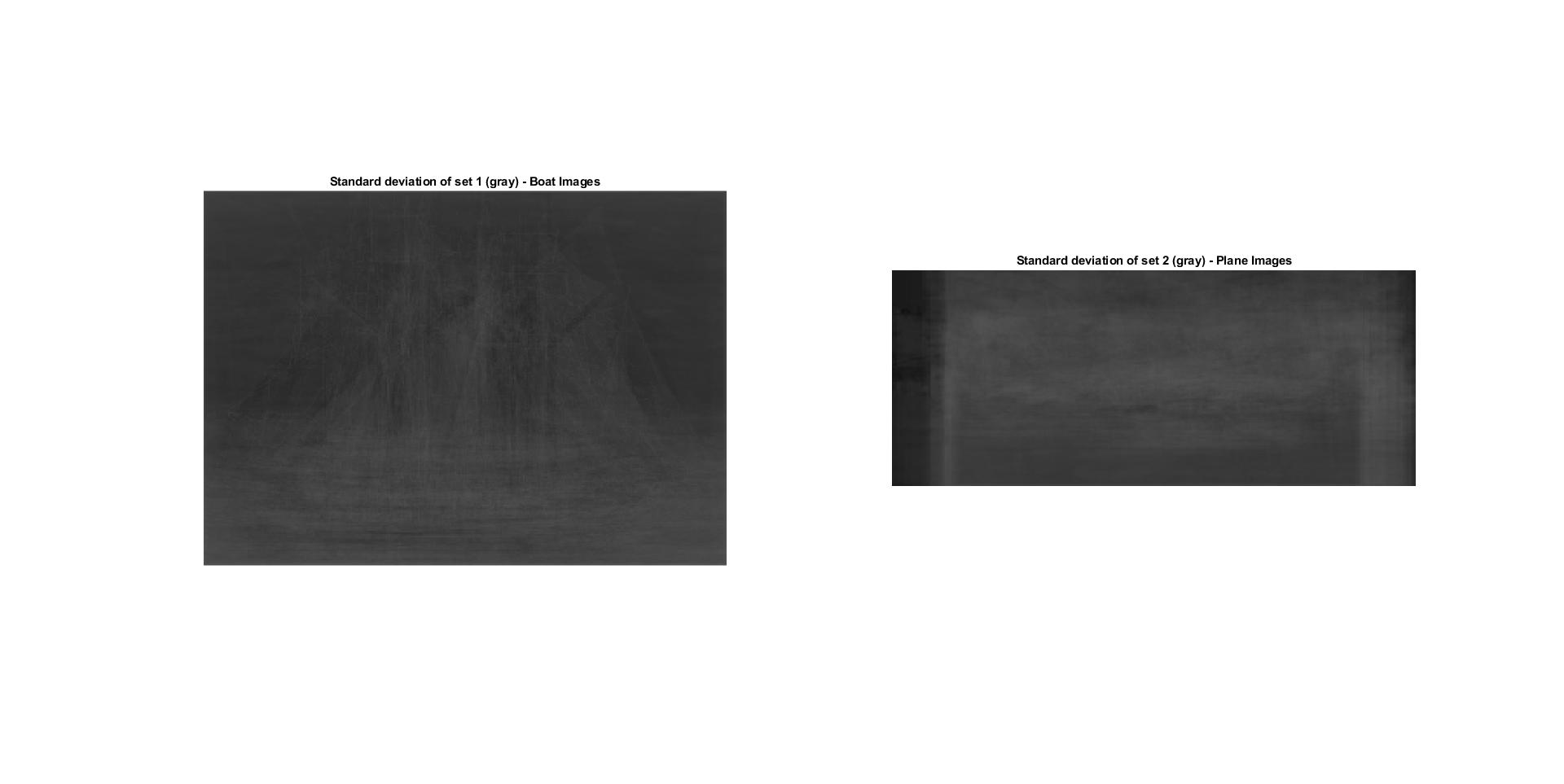
  
Sample figure with a threshold of 128  
**(d) Generate a new image (matrix), which is the same as A, but with A’s mean intensity value subtracted from each pixel. Set any negative values to 0.**Code:  
figure;

c = img - mean(mean(A)); % A is uint8, so negative values are set to 0  
imshow(c);  
  
Output:  
  
**(e) Use rand to write a function that returns the roll of a six-sided die.**Create rollDice.m with following code snippet.  
function result = rollDice()  
 result = randi(6, 1); % randi returns 1 X 1 matrix with values upto 6  
end  
And call rollDice() from the question1.m file.  
Here, randi(6, 1) suggests to return matrix with size 1\*1 and any element value up to 6.  
**(f) Let y be the vector: y = [1:6]. Use the reshape command to form a new matrix z whose first column is [1, 2, 3]’, and whose second column is [4, 5, 6]’.**Code:  
y = [1:6];  
z = reshape(y, 3, 2);  
  
Output:  
z = 1 4  
 2 5  
 3 6  
**(g) Use the min and find functions to set x to the single minimum value that occurs in A, and set r to the row it occurs in and c to the column it occurs in.**If goal is to find minimum value that does not repeat in matrix A, below command works fine.  
x = min(unique(A));  
else we can use min command as mentioned below.  
x = min(min(A));  
Once we find minimum value from matrix, we can find row and column as mentioned here.  
[r, c] = find(A == x);  
**(h) Let v be the vector: v = [1 8 8 2 1 3 9 8]. Use the unique function to compute the total number of unique values that occur in v.**v = [1, 8, 8, 2, 1, 3, 9, 8];  
numUniqValue = size(unique(v), 2);

* 1. **Programming problem: averaging images [40%]  
     Write a program that will average a collection of images, compute the standard deviations at each pixel, and display the results.  
     The images below give some examples that were generated by averaging “100 unique commemorative photographs culled from the internet” by Jason Salavon. Your program will do something similar.  
       
     There are two collects of images provided for this assignment: set1 and set2. Notice that they are all the same size within a single set.  
     Write code to do these things per set of images:  
     • Compute the average image in grayscale.  
     • Compute the average image in color, by averaging per RGB channel.  
     • Compute a matrix holding the grayscale images’ standard deviation at each pixel (i.e., X(i, j) holds the standard deviation across all the images’ gray pixel intensities at row i, column j).  
     • Display each of the above.**   
     **Apply these steps to the two sets, separately. In your write-up, briefly explain the results – why do they look the way they do?**

**Solution:**The following two functions are created in averagingImage.m and standardDeviationImages.m script files.

*averagingImage* – returns an image which is an average of images that exists in the given path. It also takes an argument whether RGB image or grayscale image is to be considered.

*standardDeviationImages* – returns an image by calculating the standard deviation of images that exists in the given path.  
Screenshots of the output images are as follows:  
1) Average of both the types of images i.e. boats and planes:  
  
2) Standard Deviation of both the types of images i.e. boats and planes:  
The program starts by specifying the paths for both the image sets. Then, it calls the function averagingImage to get the resultant output grayscale as well as RGB images. After getting the averaged images, it calls the function standardDeviationImages to get the resultant output standard deviation image. The main working of the program is inside both the functions.   
  
The averagingImage takes the input images one by one in a loop and calculates the average of the pixel values in all the images and returns the resultant output image. By averaging the images, the features of the boat and planes that are common i.e. having similar types of gray level values result in an output image having similar gray level values for the common pixels. As a result, we can see a rough structure of the boat and ship in the output image. In case of RGB images, the three channels are averaged separately in the output image, so we can see a colored image with a rough structure in the output.   
  
The standardDeviationImages works similar to the averagingImage function but instead of calculating only the average of all the images, it calculates the standard deviation of the images. By calculating the standard deviation of the images, we get the deviation i.e. how far the pixels are from the average gray level values w.r.t. the input images. This results in a rough figure of the boats and planes because all the images have common features i.e. gray levels in them which also results in a similar type of pixel values in the output image.  
  
Thus, we can see sail and the base of the boats, the wings and rudder of the plane in the output images.