Ayishwarya Narasimhan
RED ID : 818473715

CS 696 - Applied Computer Vision
HA - 1

PART I :

2) Commands, Responses and Explanation:

a) >> x = randperm(5);

Returns a vector 'x' containing a random permutation of integers 1 to 5 in random order.

```
x = 1   5   4   2   3
```

b) **>> a = [1:10];**

Returns a vector 'a' values 1 to 10 in chronological order.

```
a = 1     2     3     4     5     6     7     8     9    10
```

>> b = a([1:3:end]);

Copy to vector 'b' values from vector 'a' starting from index 1 to end in increments of 3 indices

```
b = 1     4     7    10
```

c) **>> f = [1501:2000];**

Returns a vector 'f ' with values from 1501 to 2000

>> g = find(f > 1850);

Find indices that are greater than 1850 in 'f' and store the indices in 'g'

```
>> h = f(g);
```

Copies into 'h' only those values > 1850 from 'f'

```
h  =  Columns 1 through 7
        1851       1852       1853       1854       1855       1856       1857 …
  …. Columns 148 through 150
       1998       1999       2000
```

d)
```
>> x = 22.*ones(1,10);
```

Returns a vector 'x' with 10 ones multiplied by 22 resulting with ten 22s.

```
x  = 22    22    22    22    22    22    22    22    22    22
```

```
>> y = sum(x);
```

Returns a vector 'y' with the sum of all the values in x.

```
y  = 220
```

e)
```
>> a = [1:1000];
```

Assign values 1 to 1000 to 'a'

```
>> b = a([end:-1:1]);
```

Copies into b values from a in reverse order

```
a  = Columns 1 through 7
      1          2          3          4          5          6          7  …
   …Columns 995 through 1000
      995        996        997        998        999        1000

b = Columns 1 through 7
      1000       999        998        997        996        995        994 …
  … Columns 995 through 1000
     6          5          4          3          2          1
```

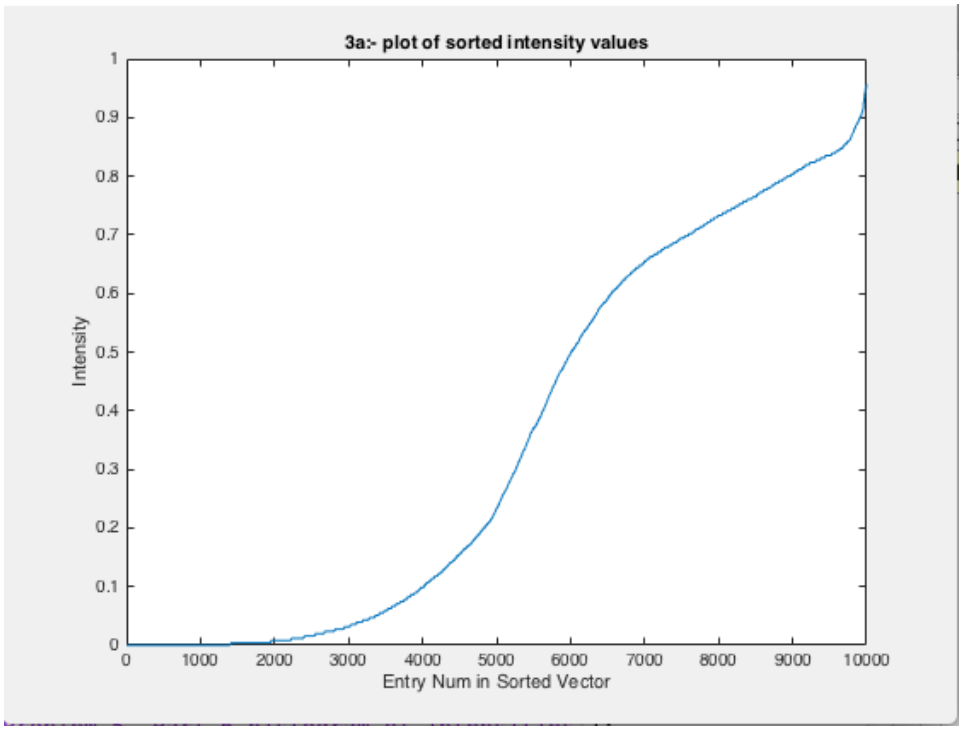
3) original image : *'100x100image.jpg'*

Code to convert original image into grayscale and save it under the name *'grayscaleImage.jpg'*

```
>> A = rgb2gray(imread('100x100image.jpg'));
>> A = im2double(X);
>> imwrite(A,'grayscaleImage.jpg','JPEG');
```

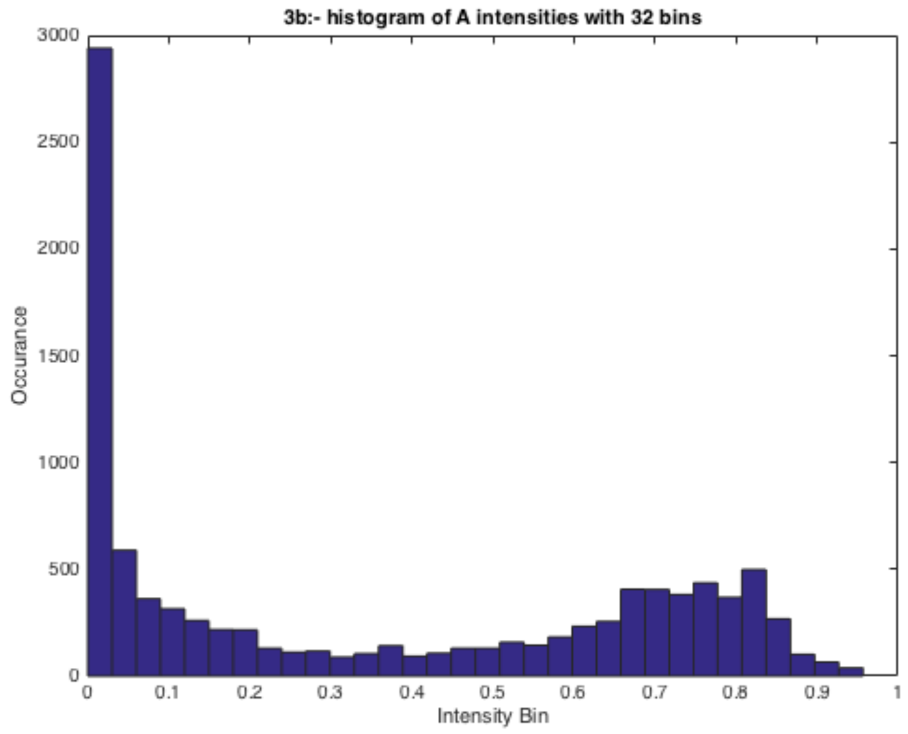
a) Sort all the intensities in A, put the result in a single 10,000-dimensional vector x, and plot the values in x.

```
>> intensiesOfA =  A( : );
>> x = sort(intensiesOfA);
>> figure
>> plot(x);
>> xlabel('Entry Num in Sorted Vector');
>> ylabel('Intensity');
>> title('3a:- plot of sorted intensity values');
```



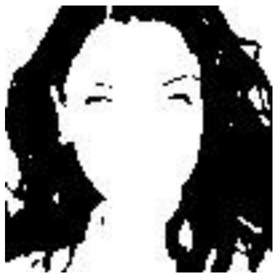
b) Display a figure showing a histogram of A's intensities with 32 bins.

```
>> figure
>> hist(intensitiesOfA,32);
>> xlabel('Intensity Bin');
>> ylabel('Occurance');
>> title('3b:- histogram of A 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.

```
>> threshold = median(intensitiesOfA);
>> binaryA = zeros(size(A));
>> binaryA(A > threshold) = 1;
>> imwrite(binaryA,'thresholdImage.jpg','JPEG');
```



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.

```
>> meanIntensity = mean(intensitiesOfA);
>> meanASubtracted = A-meanIntensity;
>> meanASubtracted( meanASubtracted < 0) = 0;
>> imwrite(meanASubtracted,'meanImage.jpg','JPEG');
```

e) Use rand to write a function that returns the roll of a six-sided die.

Function name : <u>*diceRoll.m*</u>

```
function [ result ] = diceRoll()

% Returns a roll of a six-sided die in random

result = floor(rand(1)*6) + 1;

end
```

Trial Execution #1 :

```
>> diceRoll()

   ans = 1
```

Trial Execution # 2:

```
>> diceRoll()

   ans = 5
```

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]'.

```
>> y = [1 2; 3 4; 5 6];
>> z = reshape(y,3,2);
>> z
    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.

```
 >> x = min(intensitiesOfA);
 >> [r,c] = find(A==x,1);
 >> r
      r = 35
>> c
      c = 3
```

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];
>> sizeOfUniqueV = size(unique(v));
>> numOfUniqueV = sizeOfUniqueV(2);
>> numOfUniqueV
     numOfUniqueV = 5
```

PART II :

Problem 4:

getImageSetData_*(helper function:)*

```
It gets the multi-dimensional array of color image data and grayscale
image data. All images must be same size.
Input - dirname: name of directory to search
        fileformat: file format string for images to put into set
Output - colorImages: color data array
         grayscaleImages: grayscale data array
```

a) *Average Image in Grayscale:*

writeAverageGrayscaleImage:

```
This function writes and displays the average grayscale image
  Input
      dirname: name of directory to search
      fileformat: file format string for images to put into set
      outputImageName: name to use when writing output file
```

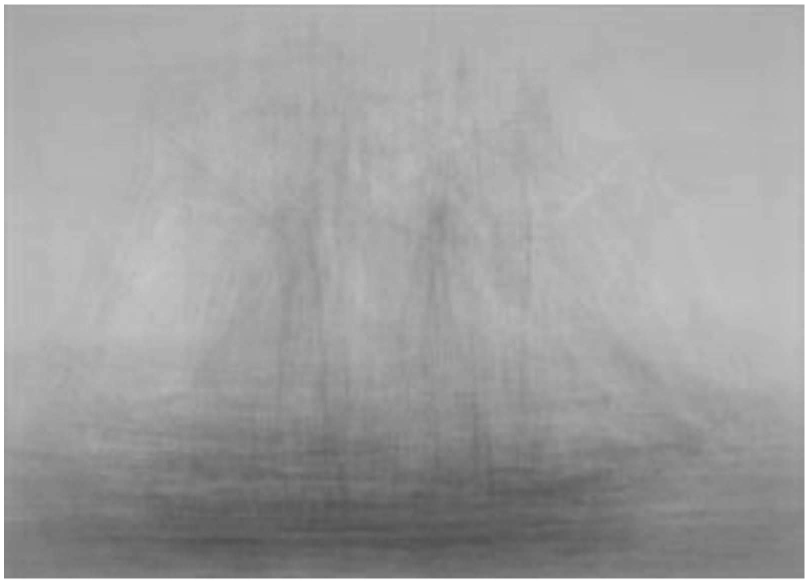


Fig 4.1a: Average Grayscale Image- Set 1

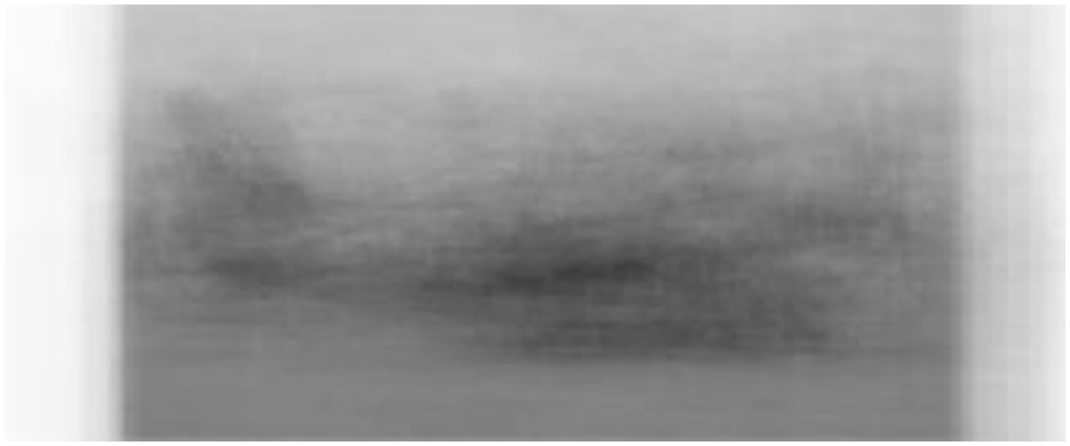


Fig 4.2a: Average Grayscale Image - Set 2

b) <u>*Average Image in Color:*</u>

writeAverageColorImage

```
This function writes and displays the average color image
   Input
      dirname: name of directory to search
      fileformat: file format string for images to put into set
      outputImageName: name to use when writing output file
```

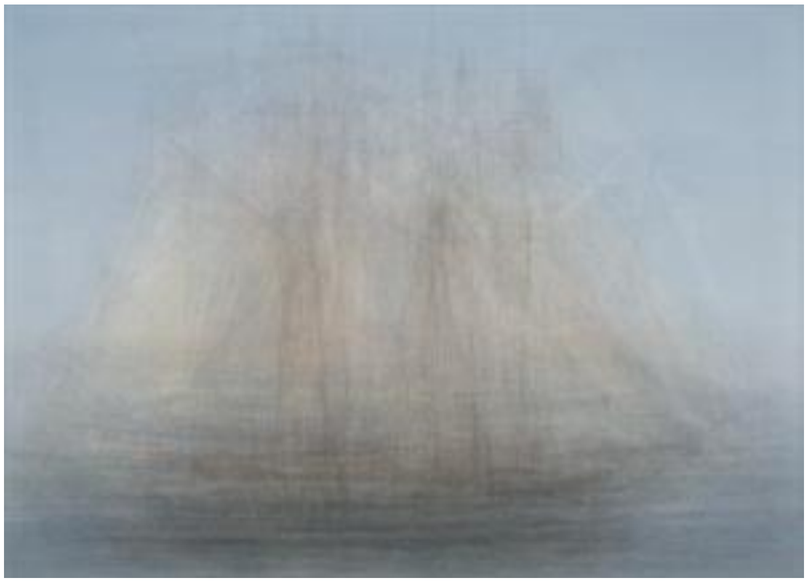


Fig 4.1b: Average Color Image- Set 1

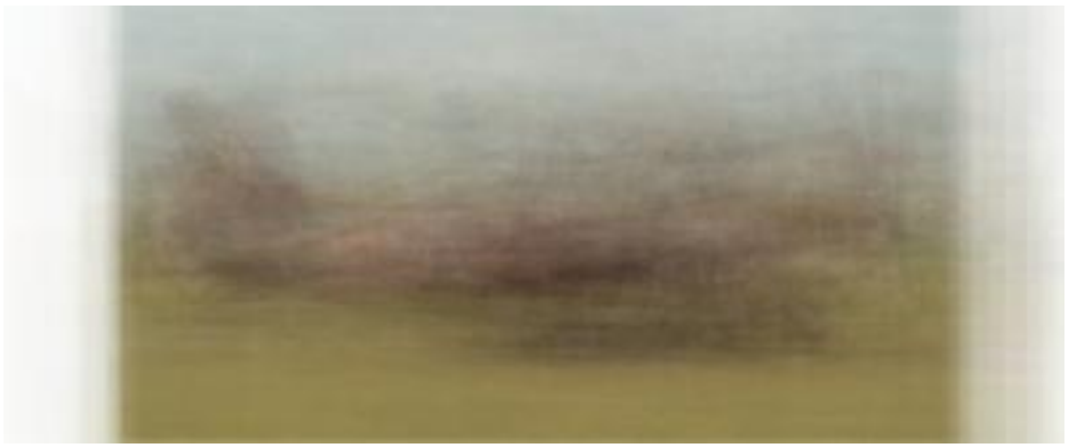


Fig 4.2b: Average Color Image- Set 2

c) *Standard Deviation Image:*

writeStndDevImage :

```
This function writes and displays the standard deviation image
 Input
     dirname: name of directory to search
     fileformat: file format string for images to put into set
     outputImageName: name to use when writing output file
```

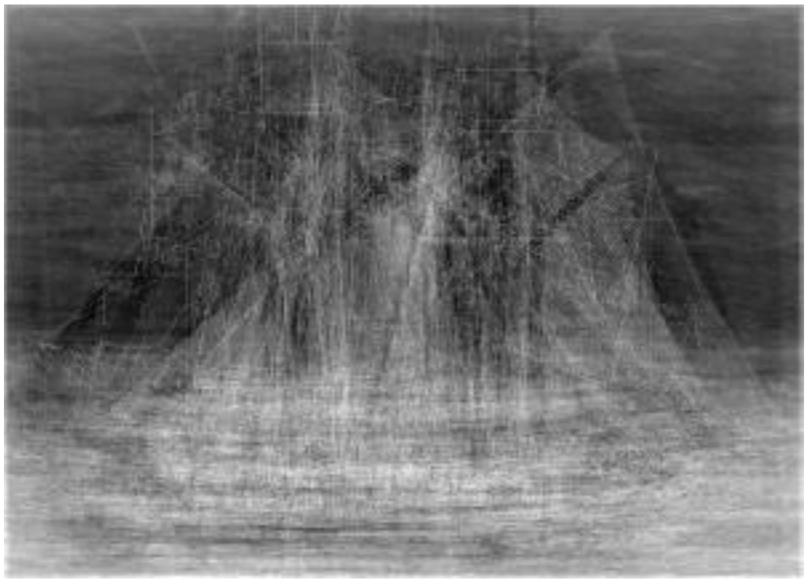


Fig 4.1.1c: Standard Deviation Grayscale Image- Set 1(above)
Fig 4.1.2c: Standard Deviation Image with colorbar- Set 1(below)

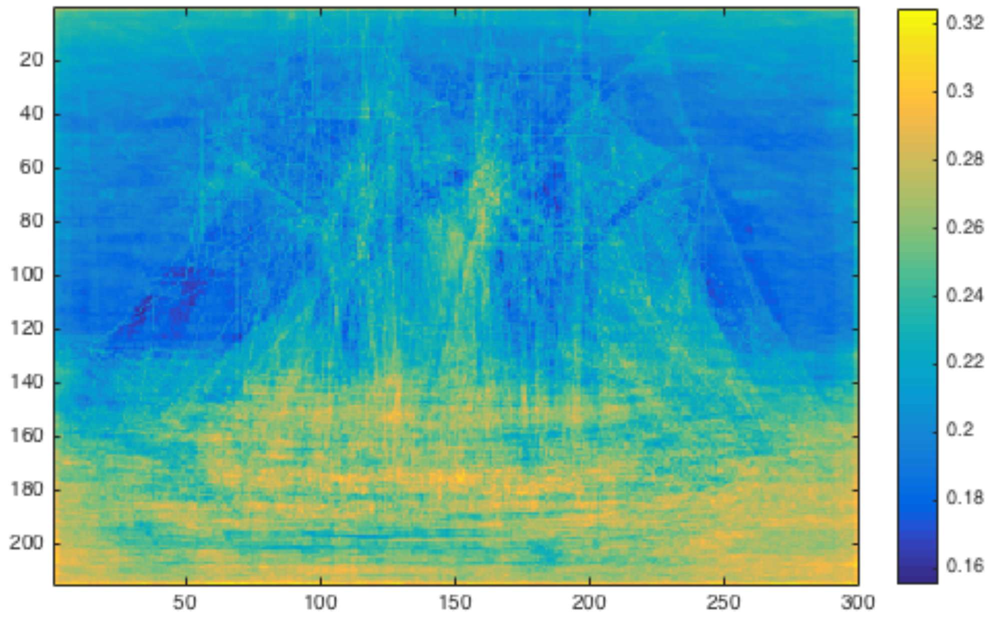
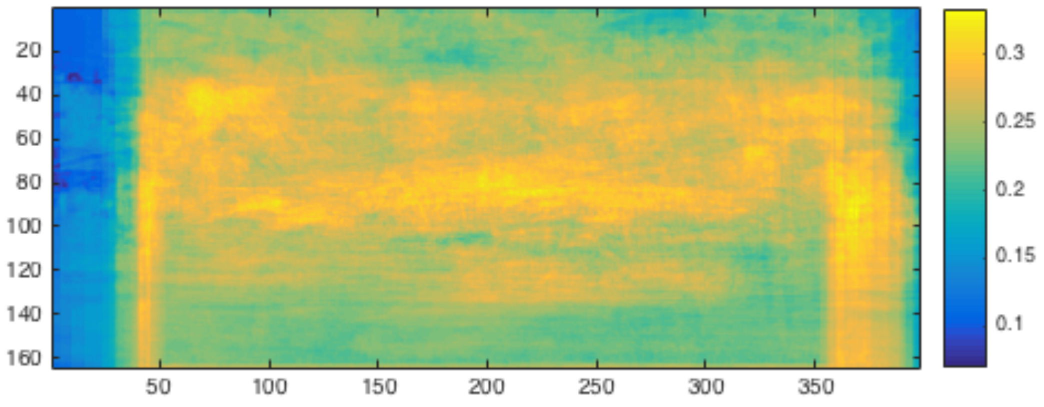


Fig 4.2c: Standard Deviation Grayscale Image- Set 2(above)
Fig 4.2.2c: Standard Deviation Image with color bar- Set 2(below)



<u>Explanation:</u>

<u>The images in set1</u> consists of sailing ships and they all look relatively similar. Thus, the average grayscale image and the average color image looks almost like a sailing ship. The structure of the sailing ship is maintained in both these images. Compared with the grayscale image, the image is easier to notice in the color image. This is due to presence of more detail in it. The standard deviation image is darker in the background and lighter in the area of the ship. This is mainly because that the background does not vary as much as the variation of the sailing ship.

<u>The images in set2</u> consists of planes. Here, in this set, there is variation in both background as well as the plane types. Thus, the average grayscale image and average color image looks very vague. Comparatively, color image looks more relatively like a plane considering more detail than the grayscale image. Due to the variation in both the background as well as the plane types, the standard deviation looks very blurry and does not look like a plane. In the standard deviation image we can observe that there is more variance in the center as to the edges. This is because, the planes are aligned more towards the center of the image.