Kuan-Ping Chang HW1 Write UP
71018021
CS116
Warm Up

3a:

The command a = [5:15]; creates an array of int-type value starting from 5 and ends at 15. The array b is then assigned values from “a” starting from the first value and then every value that is increment by 3 from the previous value until the end. So the result from b contains 5, 8, 11, and 14.

3b:

f is assigned an array of int-type value ranging from 1501 to 2000 inclusive. The next assignment to g uses a function that returns a range of values (150 total) that is greater than 1850. Finally when assigning f(g) to h, h is going to receive the range of values that are found in g. So the array in h is going to have value [1851:2000]

3c:

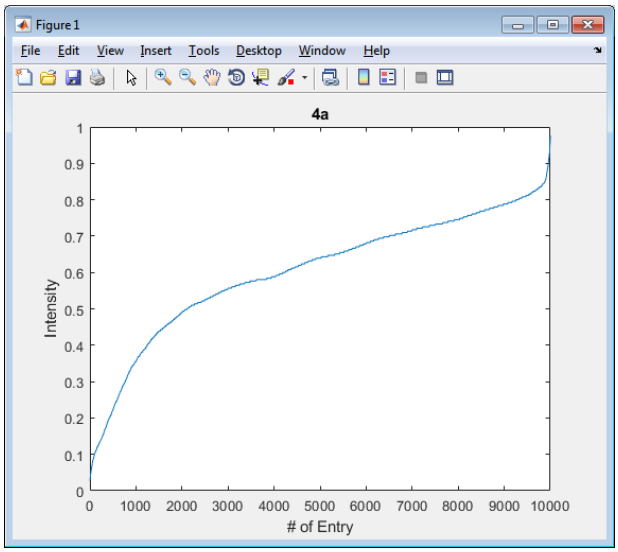
For assigning to x, function ones (1, 10) is going to create a size 10 array of all ones. When multiply by 22 it will create an array of all 22. Next we sum up all the values in the array and come down to 22 * 10 = 220 and assign it to y in y = sum(x);

3d:

First ‘a’ is assigned an array of values ranging from 1 to 100. Next ‘b’ is assigned ‘a’ with all the ordering in ‘a’ reversed. So the array in b will look like this [100, 99, 98, 97,…, 1]

Part 4.

4a.

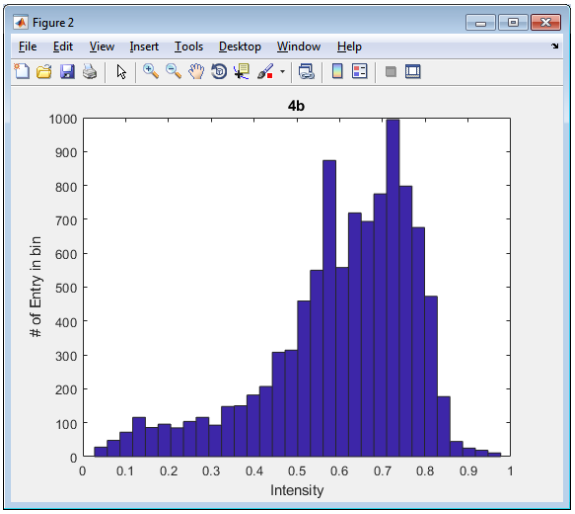



```
figure;
s = size(A);
num = s(1)*s(2);
nA = A(:);
nA = A(:)';
nA = reshape(A,[],1);
nA = reshape(A,1,[]);
x = sort(nA);
plot(x);
ylabel('Intensity');
xlabel('# of Entry');
title('4a')
```

Kuan-Ping Chang
71018021
CS116

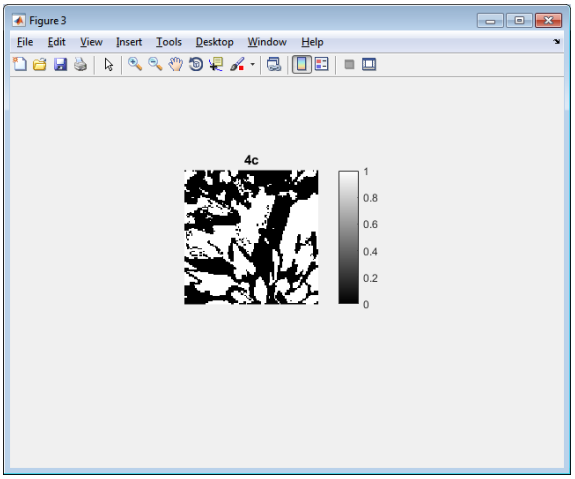
HW1 Write UP

4b.



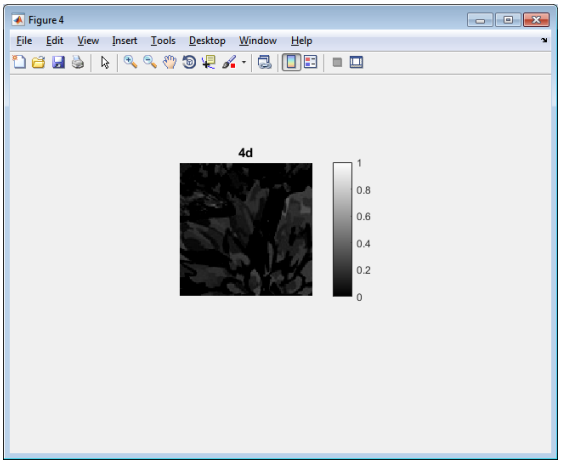

```
figure;
hist(x, 32);
ylabel('# of Entry in bin');
xlabel('Intensity');
title('4b');
```

4c




```
bA = zeros(s); %reading s
threshold = median(x);
bA(find(nA > threshold)) = 1;
figure, imshow(bA);
colorbar;
title('4c');
```

4d.




```
B = A - mean(x); %copy and create a
new matrix called
B(B < 0) = 0; %any value below zero
is set to zero
figure
imshow(B);
colorbar;
title('4d');
```

Kuan-Ping Chang HW1 Write UP
71018021
CS116
4e.

```
y = 1:6;
z = reshape(y, 3, 2);
```

Input: y = [1 2 3 4 5 6]
Output: reshape takes y and shape it into 2D matrix format that looks like [1, 4; 2, 5; 3, 6] because we specifically set 3 row and 2 column. So as result, [1 2 3] are in column 1 and [4 5 6] are in column 2.

4f.

```
x = min(min(A));
[r,c]= find(A == x);
```

Input: matrix A.

Output: Using min(A) solely will return the min value from each column. So to find the smallest value possible and other min(x) would have to be used to find the min from the found min values acquired from each column.
As result r stores row [1 1 2 2] and c stores column [49 50 50 53] of which these min values were found.

4g.

```
v = [1 8 8 2 1 3 9 8]
sz = size(unique(v));
total_number = sz(2);
```

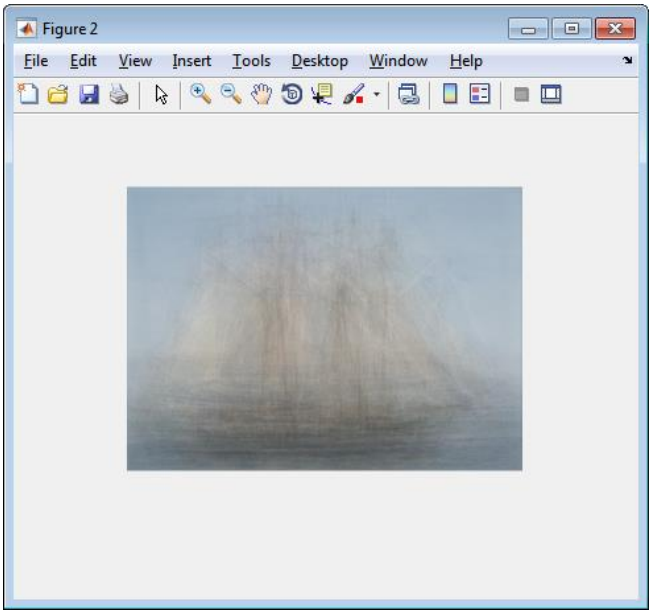
Input: v array
output: total_number = 5, unique function searches for all the unique value in the array and returns the number of unique back in matrix form

Kuan-Ping Chang
71018021
CS116

HW1 Write UP

Part 5.

Set1.

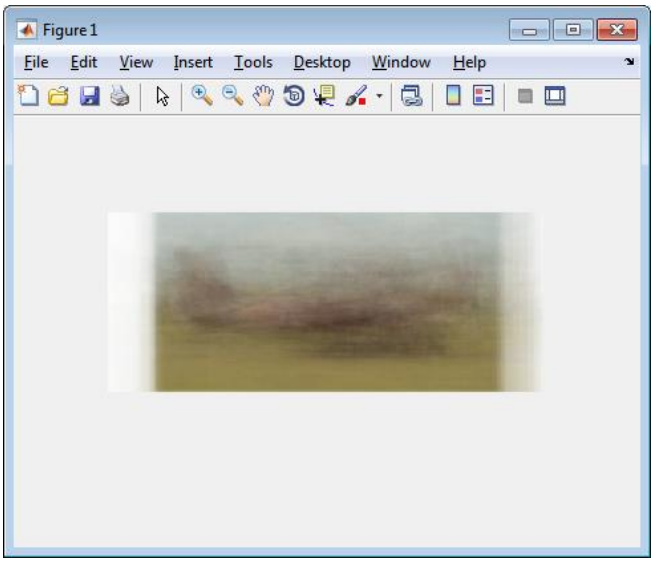


The background images are mostly composed of white or blue mixtures which can be easily averaged to create an image that distinguishes the sea and the sky.

The bodies of our ship images are mostly dark color or at least has coloration that distinguish itself fairly well in comparison to the sea and the sky. As result the average does give us a good look at the ship body.

The Sails are usually white and is usually the lightest color in the image. The average creates a good indicator of where the positions of all the sails are and gives us this result.

Set2.



The result look like this because there are “Spaces” left on the sides which weren’t clean cropped off before going into averaging. As result those white region (especially the left side) takes a huge amount of white color [1,1,1]. As for the focus of the image the “plane” body from each image are all oriented in the same direction with similar features. The ones without front engine rotator blade are added the rotator blade average those making the averaged result look like the plan has a front engine rotator.

Kuan-Ping Chang
71018021
CS116

HW1 Write UP

Part 6.

Before

After

Write Up:

The correctly demosaiced image will appear darker than the JPG because we are only adding the “linear RGB values” to make it gain color and look neutral. The picture can look better if we are able to implement white balance and color corrections.