CS 6476: Computer Vision, Fall 2019

PS0

Submitted by: Disha Das

GTID: 903542819

Date: 08/24/2019

1A Short answer problems: Using Matlab

2. Describe (in words) the result of each of the following Matlab commands.

(a) `» x = randperm(1000);`

Ans: `randperm(1000)` returns a row vector of 1000 elements (size 1x1000) that contains a random permutation of numbers between 1 and 1000 (both inclusive). This row vector is stored in **x**. The function `randperm(1000)` would return a different row vector (consisting of random permutation of numbers) everytime it is called.

(b)
```
» a = [1,2,3; 4 5 6; 7 8 9];
» b = a(2,:);
```

Ans: **a** matrix is given as:

1 2 3
4 5 6
7 8 9

`b= a(2,: )` returns the second row of **a**. This vector is stored in **b**. `a(2,:)` denotes 'every element (given by the sign ':') of 2nd row (given by the first parameter).

Thus, b = [4 5 6]

(c)
```
» a = [1,2,3; 4 5 6; 7 8 9];
» b = a(:);
```

Ans: `a(:)` takes every column of **a** and stacks them up, thereby, returning a matrix of size 9x1. This vector is stored in **b**.

Thus, b = $[1\ 4\ 7\ 2\ 5\ 8\ 3\ 6\ 9]^T$

(d)
```
» f = randn(5,1);
» g = f(find(f > 0));
```

Ans: `randn(5,1)` generates a 5x1 vector matrix consisting of random standard normally distributed real numbers (That is, numbers within distribution having Mean=0 and Standard deviation=1).

For example, f = $[-0.4229, -1.0531, 0.6478, -0.3176, 1.7690]^T$

The function `find(f>0)` returns a column vector of indices from **f** that contain numbers that are greater than 0.

Following the above example, `find(f>0)` returns `[3, 5]`^{T}

Finally, `f(find(f>0))` applies the indices obtained from the above step to **f** matrix, returning the numbers contained at those particular indices. This vector is stored in **g**.

f([3, 5]) = $[0.6478, 1.7690]^T$

g = $[0.6478, 1.7690]^T$

(e)
```
» x = zeros(1,10)+0.5;
» y = 0.5.*ones(1,length(x));
» z = x + y;
```

Ans: `x= zeros(1,10) + 0.5` creates a 1x10 column vector consisting of zeros as elements. Then a scalar addition of 0.5 to the entire vector results in a vector of the same size consisting of 0.5 as its elements. This vector is stored in **x**.

x= [0.5, 0.5, 0.5, 0.5, 0.5, 0.5]

`Length(x)` returns the length of the largest array dimension of **x**. Here, it is simply the number of elements of **x**, i.e 10.
`Ones(1, length(x))` creates a 1x10 row vector consisting of 1 as its elements. `y= 0.5.*ones(1, length(x))` returns a 1x10 row vector consisting of 0.5 as its elements. '.*' denotes scalar multiplication whereas '*' denotes vector multiplication.

y= [0.5, 0.5, 0.5, 0.5, 0.5, 0.5]

`z= x+y` simply does an element by element addition of both the vector matrices, resulting in

z = [1, 1, 1, 1, 1, 1]

(f)
```
» a = [1:100];
» b = a([end:-1:1]);
```

Ans: `a= [1:100]` results in a row vector of size 1x100 consisting of numbers from 1 to 100 (inclusive) in ascending order.

a = [1, 2, 3, … 99, 100]

'End' denotes the last element of a vector. So here, 'end' denotes 100. `a([end: step-size:1])` returns a row vector with first element as 100 and last element as 1.
`a([end:-1:1])` has a step-size of -1. Therefore, element 2 of the vector would be 100-1=99. The 3rd element would be 99-1=98 and so on.
Therefore, `a([end:-1:1])` returns a row vector of size 1x100 consisting of numbers from 1 to 100 in a decreasing order.

b = [100, 99, 98, … 2, 1]

3. Write a few lines of code to do each of the following. Copy and paste your code into the answer sheet. [16 points]

(a) Use rand to write a function that returns the roll of a six-sided die over N trials.

Ans:

```
a = diceRoll(input('Enter number of trials: '))

function trials = diceRoll(n)
      if ~isscalar(n)
            error('Enter a scalar');
      elseif mod(n,1)~=0
            error('Enter an Integer');
      elseif n<=0
            error('Enter an integer greater than 0')
      end
      trials = floor(1 + 6*rand(n,1));
end
```

(b) Let y be the vector: y = [1 2 3 4 5 6]'. Use the reshape command to form a new matrix z that looks like this: z =

1 3 5
2 4 6

Ans:

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

(c) Use the max and find functions to set x to the maximum value that occurs in z (above), and set r to the row number it occurs in and c to the column number it occurs in.

Ans:

```
x = max(max(z));
[r,c]=find(z==x);
```

(d) Let v be the vector: v = [1 8 8 2 1 3 9 8]. Set a new variable x to be the number of 1's in the vector v.

Ans:

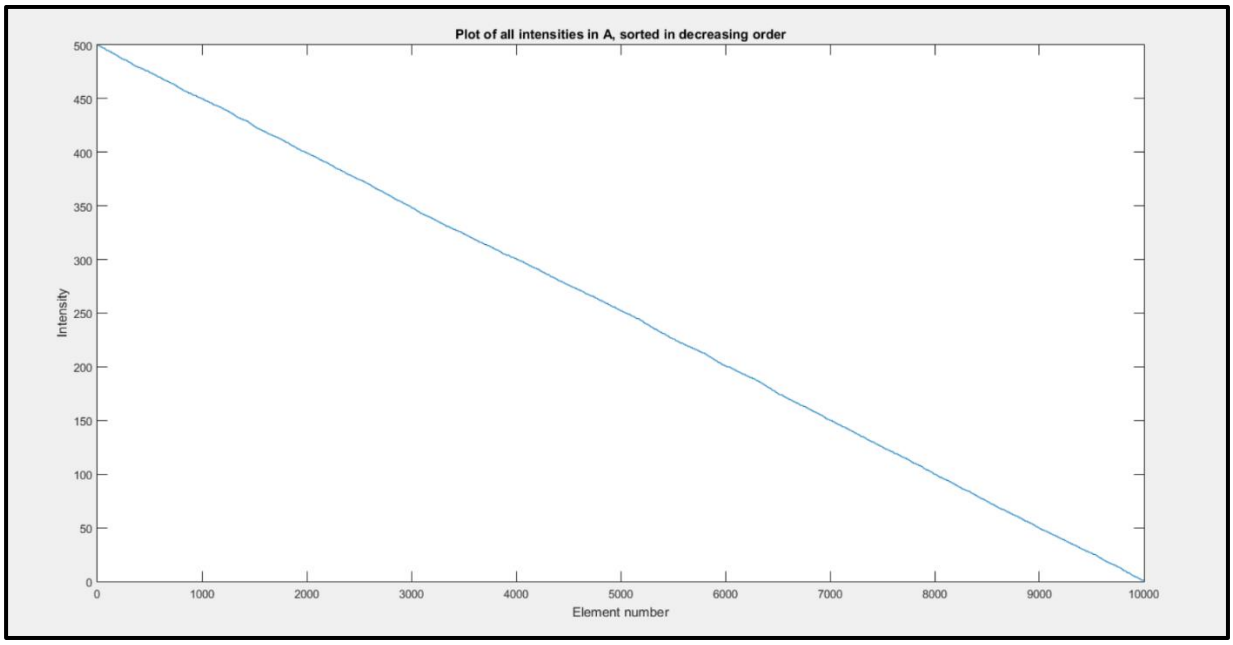
```
v = [1 8 8 2 1 3 9 8];
x = sum(v(:)==1);
```

4. Create any 100 x 100 matrix A (not all constant). Save A in a .mat file called inputAPS0Q1.mat and submit it. Write a script which loads inputAPS0Q1.mat and performs each of the following actions on

A. Name it PS0Q1.m and submit it. [30 points]

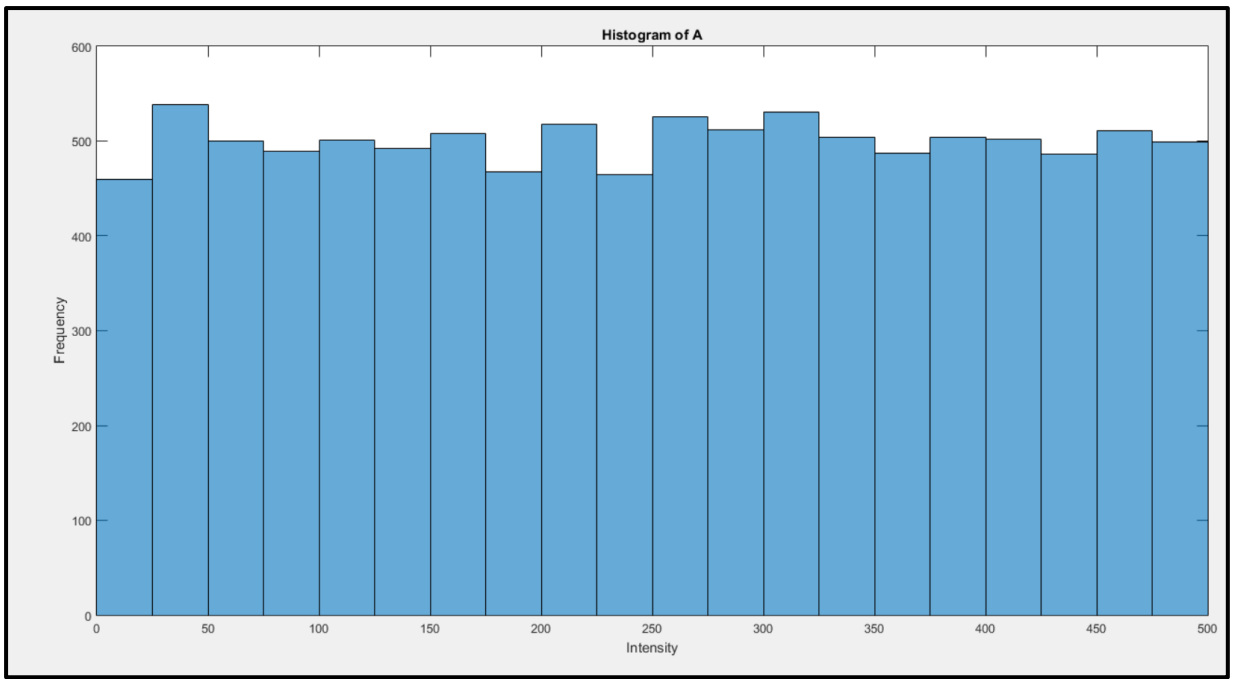
(a) Plot all the intensities in A, sorted in decreasing value. Provide the plot in your answer sheet. (Note, in this case we don't care about the 2D structure of A, we only want to sort the list of all intensities.)

Ans:



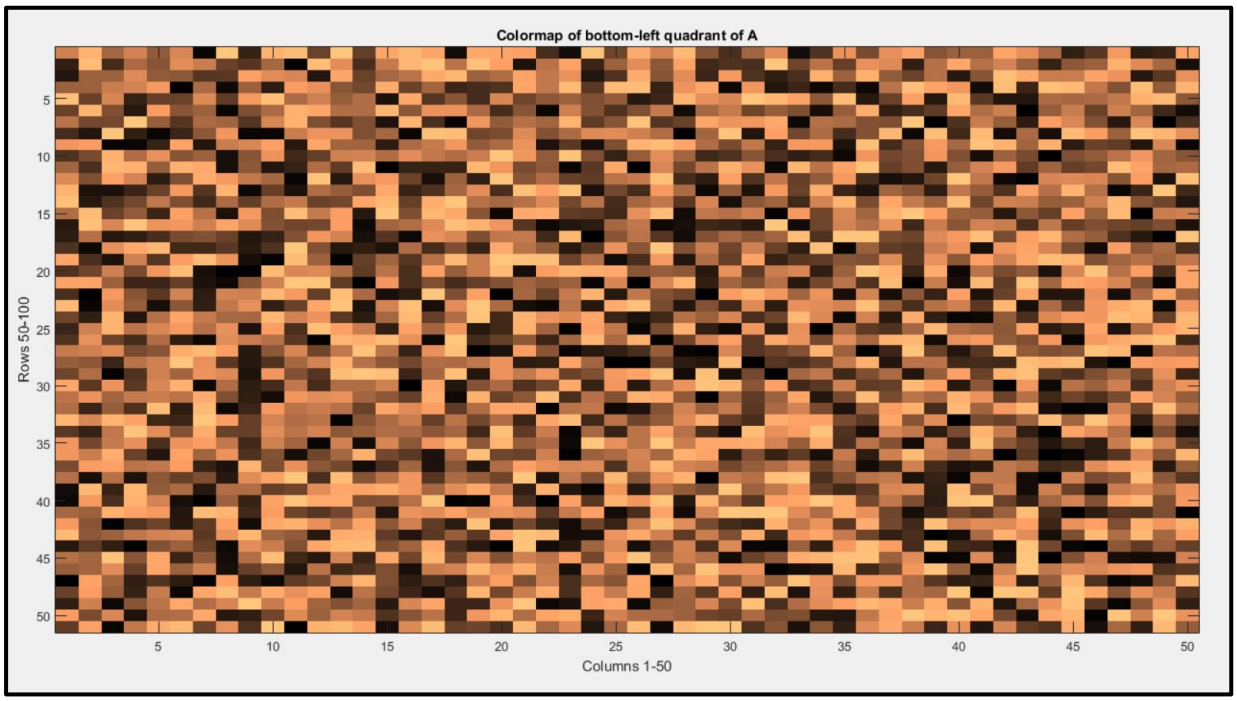
(b) Display a histogram of A's intensities with 20 bins. Again, we do not care about the 2D structure. Provide the histogram in your answer sheet.

Ans:



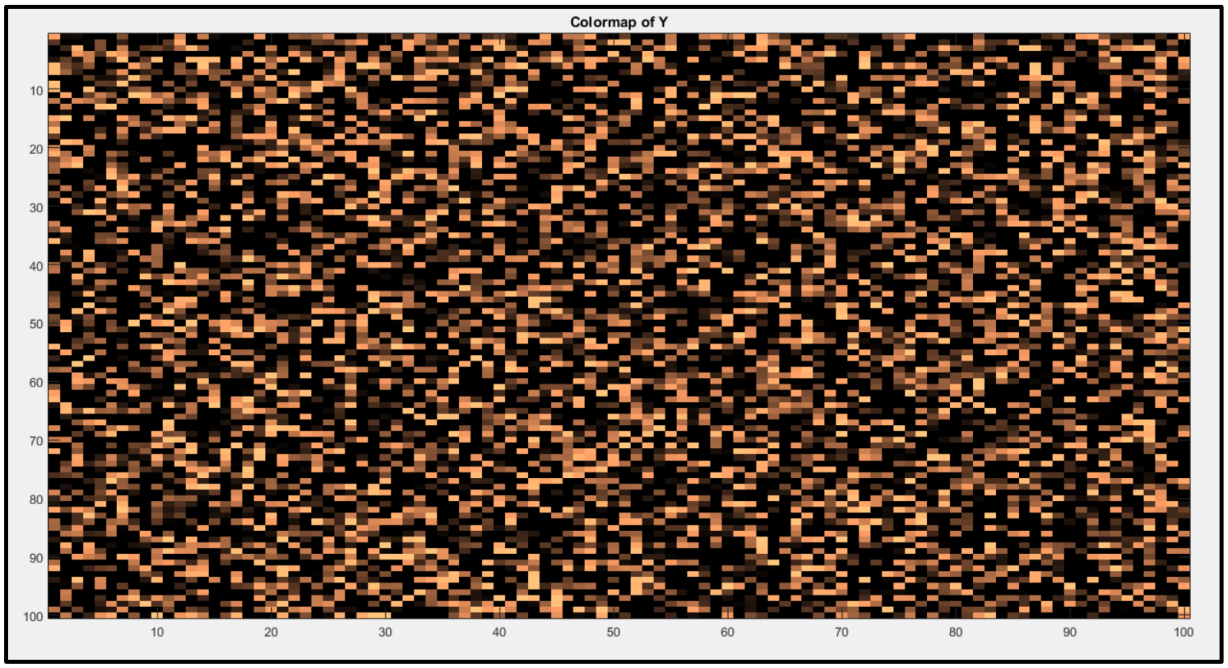
(c) Create a new matrix X that consists of the bottom left quadrant of A. Display X as an image in your answer sheet using imagesc. Look at the documentation for colormap. Try colormap gray, colormap jet, colormap copper and others. Save X in a file called outputXPS0Q1.mat and submit the file.

Ans:



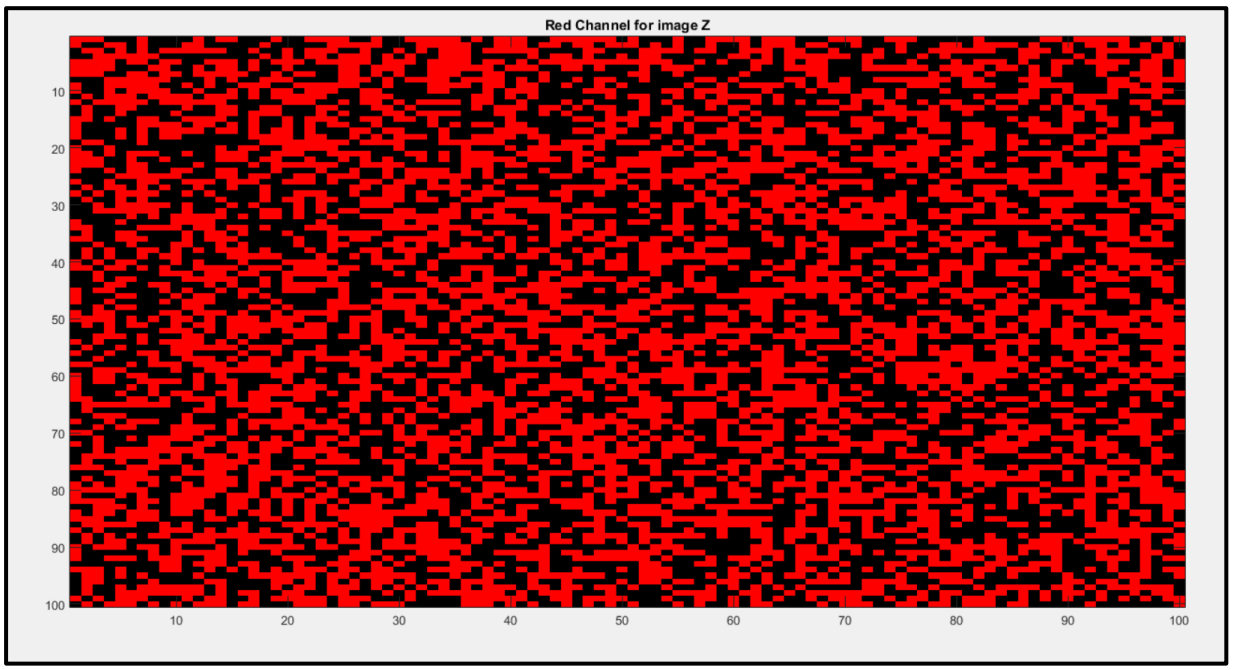
(d) Create a new matrix Y, which is the same as A, but with A's mean intensity value subtracted from each pixel. Display Y as an image in your answer sheet using imagesc. Save Y in a file called outputYPS0Q1.mat and submit the file.

Ans:



(e) Create a new matrix Z that represents a color image the same size as A, but with 3 channels to represent R, G and B values. Set the values to be red (i.e., R = 255, G = 0, B = 0) wherever the intensity in A is greater than a threshold t = the average intensity in A, and black everywhere else. Display Z as an image in your answer sheet using imagesc and imshow. Be careful with typecasting. Save Z as outputZPS0Q1.png and submit the file. Be sure to view outputZPS0Q1.png in an image viewer to make sure it looks right.

Ans:



2. Short programming example [36 points]

1. Load the input color image and swap its red and green color channels. Save the output as swapImgPS0Q2.png.
2. Convert the input color image to a grayscale image. Save the output as grayImgPS0Q2.png.
3. Perform each of the below transformations on the grayscale image produced in part 2 above.

(a) Convert the grayscale image to its negative image, in which the lightest values appear dark and vice versa. Save the output as negativeImgPS0Q2.png.
(b) Map the grayscale image to its mirror image, i.e., flipping it left to right. Save the output as mirrorImgPS0Q2.png.
(c) Average the grayscale image with its mirror image (use typecasting). Save the output as avgImgPS0Q2.png.
(d) Create a matrix N whose size is same as the grayscale image, containing random numbers in the range [0 255]. Save this matrix in a file called noise.mat(npy). Add N to the grayscale image, then clip the resulting image to have a maximum value of 255. Save the output as addNoiseImgPS0Q2.png.

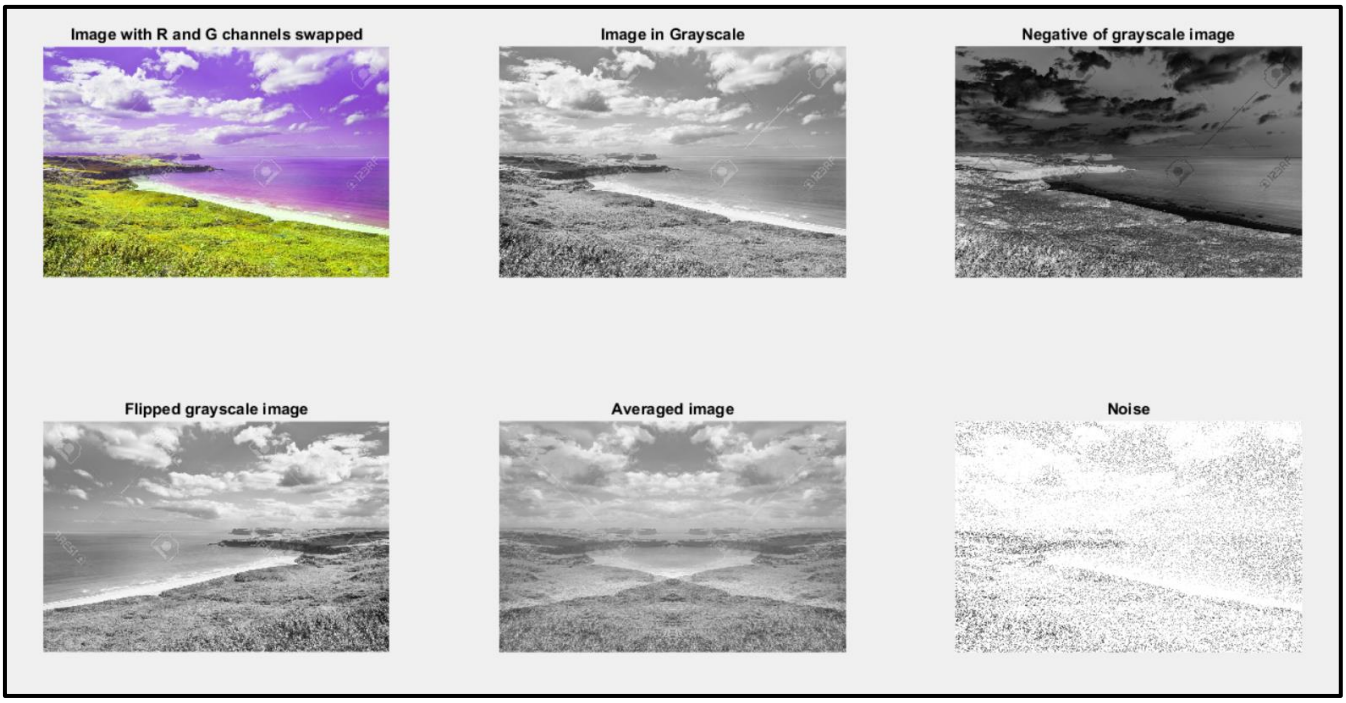


Image with R and G channels swapped
Image in Grayscale
Negative of grayscale image
Flipped grayscale image
Averaged image
Noise