ECE 4984/5554: Computer Vision, Fall 2015

PS0

Christopher Mobley

Due: Monday, 31 AUG 15

1 Short answer problems: Do either A (if you want to us MATLAB) or B (if you want to use Python) [64 points]

B Using Python

1. Read through the provided Python numpy and Matplotlib introduction code and comments: http://cs231n.github.io/python-numpy-tutorial/ or https://filebox.ece.vt.edu/~F15ECE5554ECE4984/resources/numpy.pdf. Open an interactive session in Python and test the commands by typing them at the prompt. (Skip this step if you are already familiar with Python and numpy.)
2. Describe (in words) the result of each of the following Python commands. Search the numpy API documentation http://docs.scipy.org/doc/numpy/ if needed, but try to determine the output without entering the commands into Python. Do not submit a screenshot of the result of typing these commands. [18 points]
   a. x = np.random.permutation(1000)

      np.random.permutation(1000) randomly permute np.arange(1000); so, an array of shape (1000, 0) is created with values between [0, 1000) in a random order.

   b. a = np.array([[1,2,3],[4,5,6],[7,8,9]])
      b = a[2,:]

      Np.array(([[1,2,3],[4,5,6],[7,8,9]]) creates the following array

      ```
      [[1, 2, 3]
       [4, 5, 6]
       [7, 8, 9]]
      ```

      ;while, a[2,:] slices the array pulling out the subarray included below which consists of all elements in the 3rd row.

      ```
      [7, 8, 9]
      ```

   c. a = np.array([[1,2,3],[4,5,6],[7,8,9]])
      b = a.reshape(-1)

Np.array([[1,2,3],[4,5,6],[7,8,9]]) creates the following array
[[1, 2, 3]
 [4, 5, 6]
 [7, 8, 9]]
;while, a.reshape(-1) creates a 1-D array in which the value is inferred from the length of the array and remaining dimensions. So, the following array will be create [1, 2, 3, 4, 5, 6, 7, 8, 9].

d. f = np.random.randn(5,1)
   g = f[f>0]

   np.random.rand(5,1) creates an array of shape (5, 1) composed of a random sample from a normal distribution N(0, 1); while, f[f>0] uses Boolean array indexing to create an array with shape (# of true cases,) in which all elements are greater than 0.

e. x = np.zeros(10)+0.5
   y = 0.5*np.ones(len(x))
   z = x + y

   np.zeros(10) + 0.5 creates an array of all 0 with shape (10,) and then adds 0.5 to each element. So, the resulting array would be one of all 0.5 and shape (10,). While np.ones(len(x)) calculate the length of x, which is 10. Then creates an array of all ones with a shape (10,) then multiplies each element of this array by 0.5 to create an array of all 0.5 with a shape (10,). x + y adds the two array together elementwise resulting in an array of all ones with a shape of (10,).

f. a = np.arange(1,100)
   b = a[::-1]

   np.arange(1,100) creates an array of shape (99,) filled with value between [1,100) in order; while, a[::-1] slices the array to create a subarray with step through the entire array with a step size of -1. Thus, instead of having an array of shape (99,) which goes from 1 to 99 you will have an array of the same shape expect it will go from 99 to 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 numpy.random.rand to write a function that returns the roll of a six-sided die over N trials.

```
import numpy as np

def dice_roller(N):
    return np.trunc((np.random.rand(1,N) * 6) + 1)
```

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

```
import numpy as np

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

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

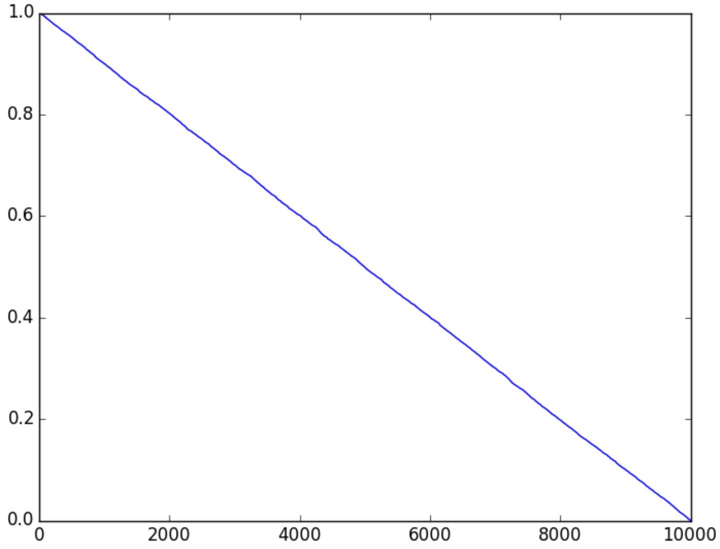
```
import numpy as np

x = np.max(z)
r = np.where(z==x)[0][0]
c = np.where(z==x)[1][0]
```

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

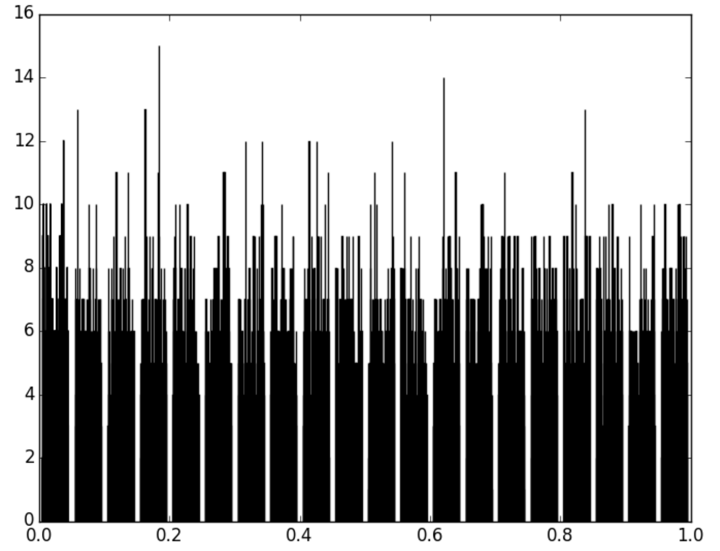
```
import numpy as np

v = np.array([1, 8, 8, 2, 1, 3, 9, 8])
x = np.sum(v[v==1])
```

4. Create any 100 x 100 matrix A (not all constant). Save A in a .npy file called inputAPS0Q1.npy and submit it. Write a script which loads inputAPS0Q1.npy and performs each of the following actions on A. Name it PS0Q1.py and submit it. [30 points]

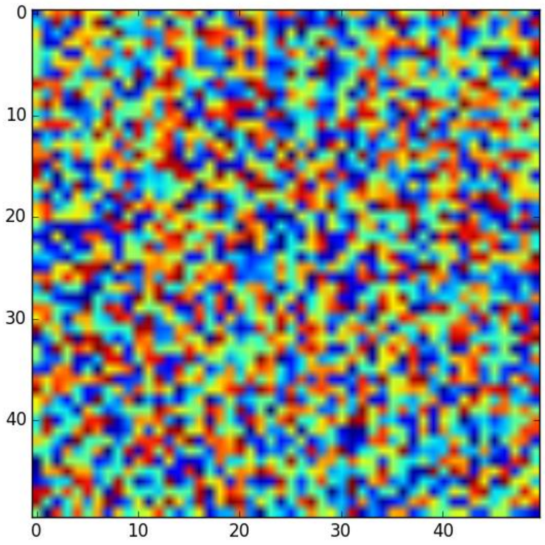
   a. Plot all the intensities in A, sorted in decreasing value. Provide the plot in your answer sheet. (Note, in this case we don't care about the 2D structure of A, we only want to sort the list of all intensities.)



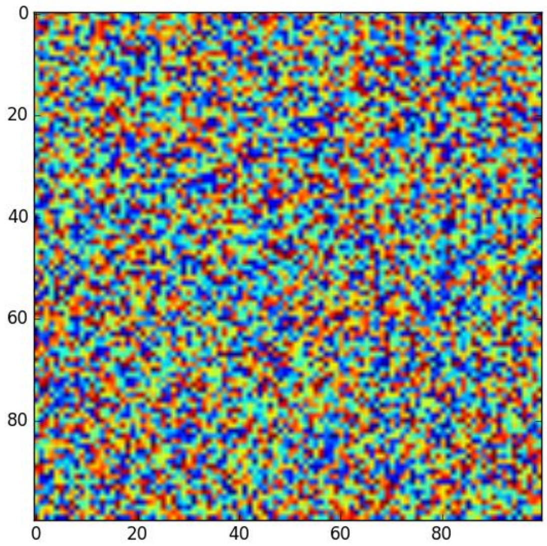
b. Display a histogram of A’s intensities with 20 bins. Again, we do not care about the 2D structure. Provide the histogram in your answer sheet.



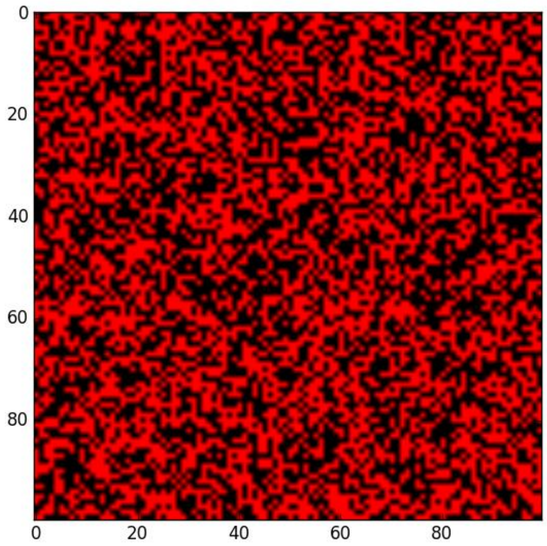
c. Create a new matrix X that consists of the bottom left quadrant of A. Display X as an image in your answer sheet using matplotlib.pyplot.imshow with no interpolation (blurry effect). Look at the documentation for matplotlib.pyplot.imshow. Save X in a file called outputXPS0Q1.npy and submit the file.



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 matplotlib.pyplot.imshow. Save Y in a file called outputYPS0Q1.npy and submit the file.



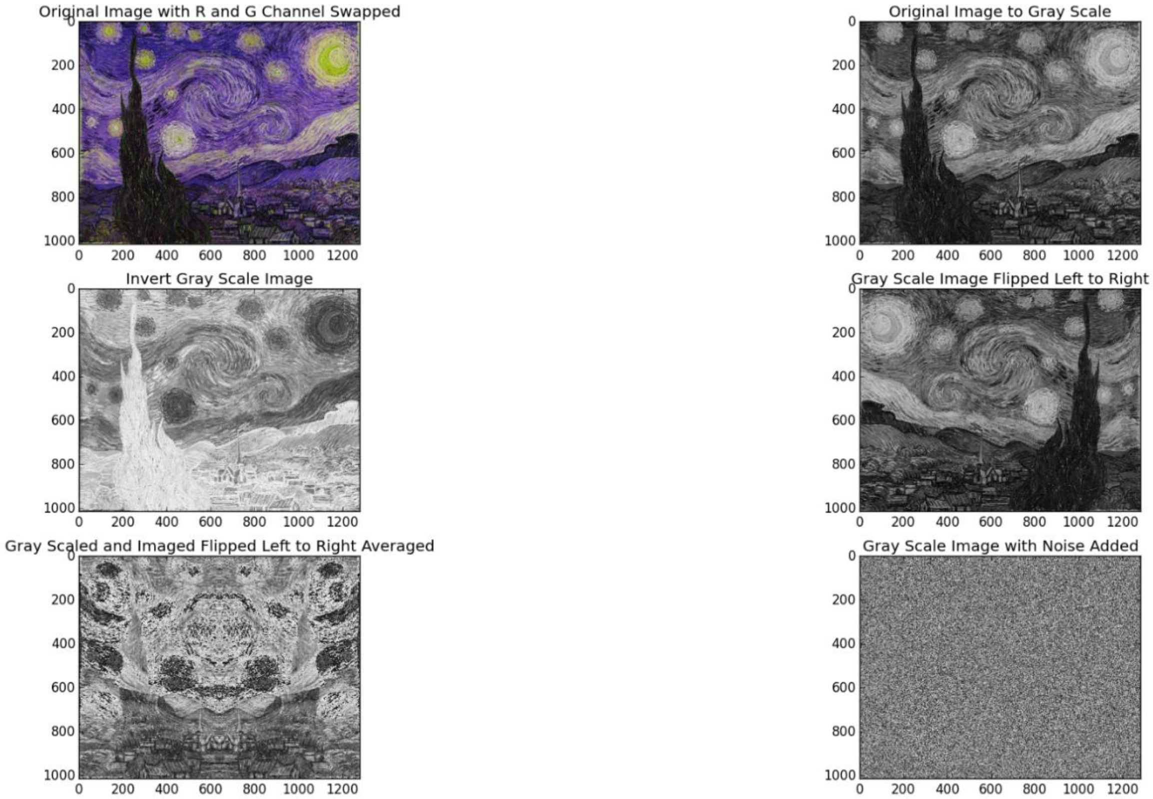
e. Create a new matrix Z that represents a color image the same size as A, but with 3 channels to represent R, G and B values. Set the values to be red (i.e., R = 1, G = 0, B = 0) wherever the intensity in A is greater than a threshold t = the average intensity in A, and black everywhere else. Display Z as an image in your answer sheet using matplotlib.pyplot.imshow. 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.



2 Short programming example (you can use MATLAB or Python, whatever you prefer) [36 points]

Choose any color image from the web or your personal collection and name it inputPS0Q2.jpg. Write a script which performs the following transformations and displays the results in a figure using the Matlab or Python subplot (matplotlib.pyplot.subplots) function in a 3x2 grid (3 rows and 2 columns). Each subplot should contain the output of each of the below operations. Label each subplot with an appropriate title. Provide the subplot in your answer sheet. Avoid using loops. Name the script PS0Q2.m(py). Note: The transformed images should be in png format.

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.


Original Image with R and G Channel Swapped
Original Image to Gray Scale
Invert Gray Scale Image
Gray Scale Image Flipped Left to Right
Gray Scaled and Imaged Flipped Left to Right Averaged
Gray Scale Image with Noise Added
0
200
400
600
800
1000
1200

Note: I discussed this assignment with Murat Ambarkutuk, Rich Fedora, Orson Lin, and Yi Tien