CS 4476: Introduction to Computer Vision, Fall 2018
PS0

Instructor: Devi Parikh

Due: Monday, August 27th, 11:59 pm

Instructions

1. Answer sheets, code and input/output images must be submitted on Canvas. Hard copies will not be accepted.
2. Save your answer sheet containing the written answers in a file named: FirstName_LastName_PS0.pdf. Please submit your code, input/output images and answer sheets in a zip file named: FirstName_LastName_PS0_mat.zip if using Matlab or FirstName_LastName_PS0_py.zip if using Python. Please do not create subdirectories within the main directory.
3. For the implementation questions, make sure your code is bug-free and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not run out of the box.
4. If plots are required, you must include them in your answer sheet (pdf) and your code must display them when run. Points will be deducted for not following this protocol.

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

A Using Matlab

1. Read through the provided Matlab introduction code and comments:
`https://filebox.ece.vt.edu/~F15ECE5554ECE4984/resources/matlab.pdf`.
Open an interactive session in Matlab and test the commands by typing them at the prompt. (Skip this step if you are already familiar with Matlab.)
2. Describe (in words) 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. [18 points]

   (a) `» x = randperm(1000);`

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

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

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

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

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

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.

   (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: $z = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$.

   (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.

   (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`.

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.)

   (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 `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.

   (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.

   (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.

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]

```
> import numpy as np
```

(a)
```
> x = np.random.permutation(1000)
```

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

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

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

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

(f)
```
> a = np.arange(1,100)
> b = a[::-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 return the roll of a six-sided die over N trials.

   (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]]`

   (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 number (0-indexed) it occurs in and `c` to the column number (0-indexed) it occurs in.

   (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`.

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`.

Be sure to submit the input image, all the output images and `noise.mat(npy)`.

Matlab Tips: Do the necessary typecasting (`uint8` and `double`) when working with or displaying the images. Some useful functions: `title`, `subplot`, `imshow`, `mean`, `imread`, `imwrite`, `rgb2gray`.

Python Tips: Do the necessary typecasting (`uint8` and `double`) when working with or displaying the images. If you can't find some functions in numpy (such as rgb2gray), you can write your own function. For example:

```
def rgb2gray(rgb):
  return np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140])
```
[1]

[1] These are the weights that the MATLAB rbg2gray function uses.
This assignment closely follows the PS0 assignment of Kristen Grauman's CS 376: Computer Vision at UT Austin