CS231A: Computer Vision, From 3D Reconstruction to Recognition Homework #0

(Winter 2022) Due: **Sunday, January 9**

Student Name: Vasu Patel, SUID: vgpatel1

This is a short tutorial on how to use Python and a review of some small linear algebra ideas. The point of this assignment is to get you used to manipulating matrices and images in Python with NumPy.

Please put together a PDF with your results for each problem, and submit it to the appropriate assignment on Gradescope. We recommend you to add these answers to the latex template files on our website, but you can also create a PDF in any other way you prefer. There will be an additional coding assignment on Gradescope that has an autograder and is there to help you double check your code. Make sure you use the provided ".py" files to write your Python code. Submit to both the PDF and code assignment, as we will be grading the PDF submissions and using the coding assignment to check your code if needed.

NOTE: This problem set is not representative of future problem sets in terms of length or difficulty, but the logistics will be similar (submission, Piazza, etc).

Most likely, you will find the future problem sets to be more difficult than this one. This problem set is simply to ensure you have a sufficient understanding of the basic prerequisites for this class.

Submit your published results to the class Gradescope (Code **6PGPN3**). Feel free to ask any questions to the class Piazza forum (but use a private post if you have code or specifics to discuss).

1 Piazza

Please register on the class Piazza forum: `piazza.com/stanford/winter2021/cs231a`.

2 Basic Matrix/Vector Manipulation (10 points)

In Python, please calculate the following. Given matrix M and vectors a,b,c such that

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 2 & 2 \end{bmatrix}, a = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, b = \begin{bmatrix} -1 \\ 2 \\ 5 \end{bmatrix}, c = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 2 \end{bmatrix}$$

(a) Define Matrix M and Vectors a,b,c in Python. One helpful Python package that is commonly used for linear algebra related problems is Numpy.

Answer Problem 2a:
M = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [0, 2, 2]])
a = np.array([1, 1, 0])
b = np.array([-1, 2, 5])
c = np.array([0, 2, 3, 2])

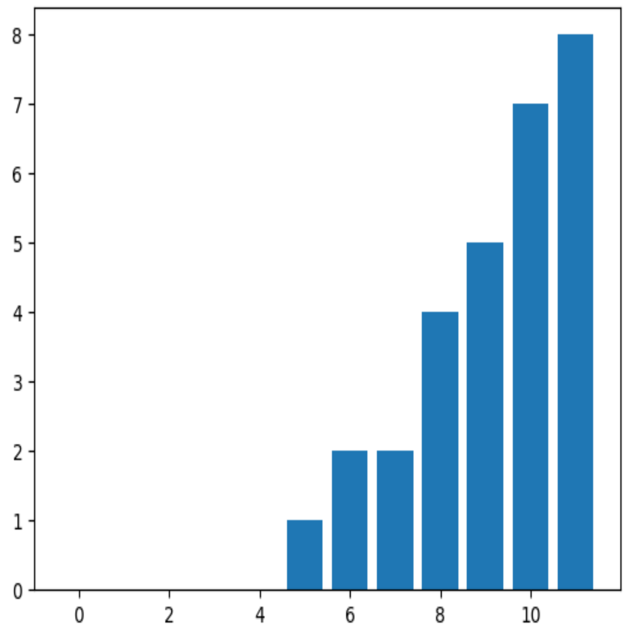
(b) Find the dot product of vectors a and b (i.e. $a^\top b$). Save this value to the variable `aDotb` and write its value in your report.
Answer Problem 2b: 1

(c) Find the element-wise product of a and b $[a_1 b_1, a_2 b_2, a_3 b_3]^T$ and write it in your report.
Answer Problem 2c: [-1 2 0]

(d) Find $(a^\top b)Ma$ and write it in your report.
Answer Problem 2d: [3 9 15 2]

(e) Without using a loop, multiply each row of M element-wise by a. (Hint: The function numpy.tile() may come in handy). Write the result in your report.
Answer Problem 2e:
[[1 2 0]
[4 5 0]
[7 8 0]
[0 2 0]]

(f) Without using a loop, sort all of the values of the new M from (e) in increasing order and plot them in your report.
Answer Problem 2f:



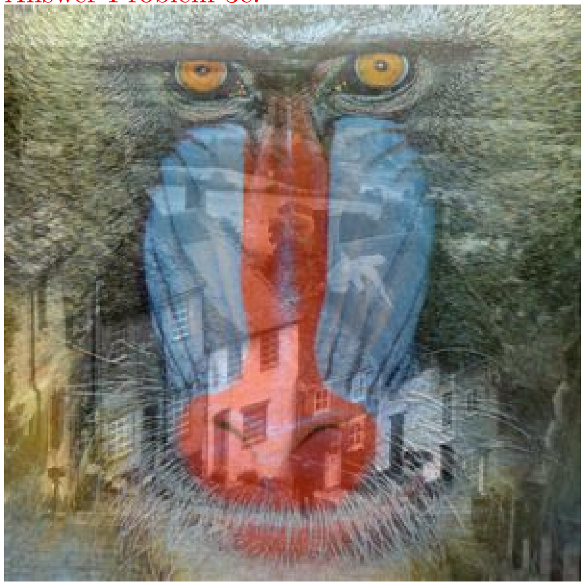
3 Basic Image Manipulations (20 points)

(a) Read in the images, image1.jpg and image2.jpg, as color images.
Answer Problem 3a:
img1 = io.imread(fname="image1.jpg")
img2 = io.imread(fname="image2.jpg")

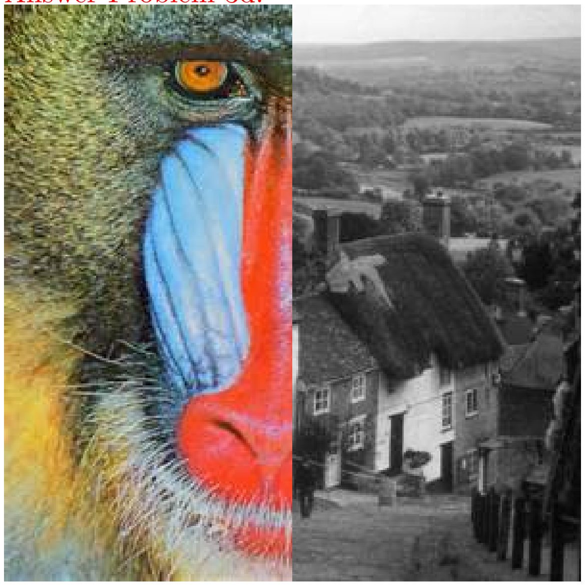
(b) Convert the images to double precision and rescale them to stretch from minimum value 0 to maximum value 1.

Answer Problem 3b:
img1 = (img1 - np.min(img1)) / (np.max(img1) - np.min(img1))
img2 = (img2 - np.min(img2)) / (np.max(img2) - np.min(img2))

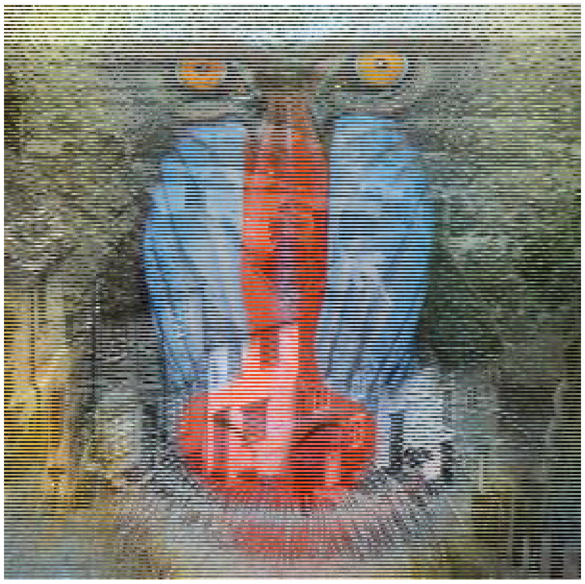
(c) Add the images together and re-normalize them to have minimum value 0 and maximum value 1. Display this image in your report.
Answer Problem 3c:



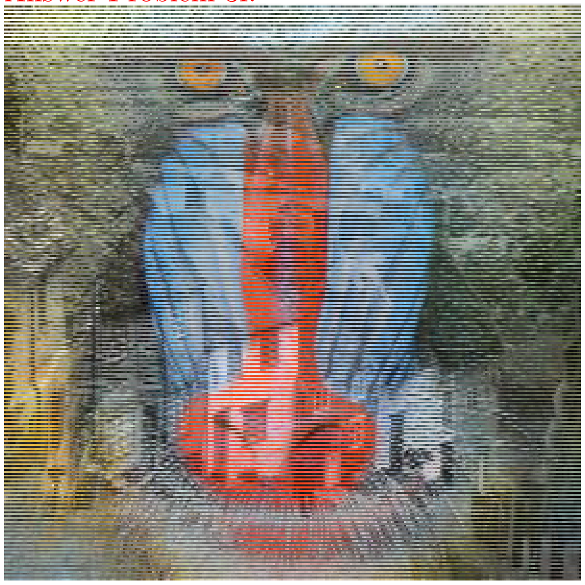
(d) Create a new image such that the left half of the image is the left half of image1 and the right half of the image is the right half of image2. Display this image in your report.
Answer Problem 3d:



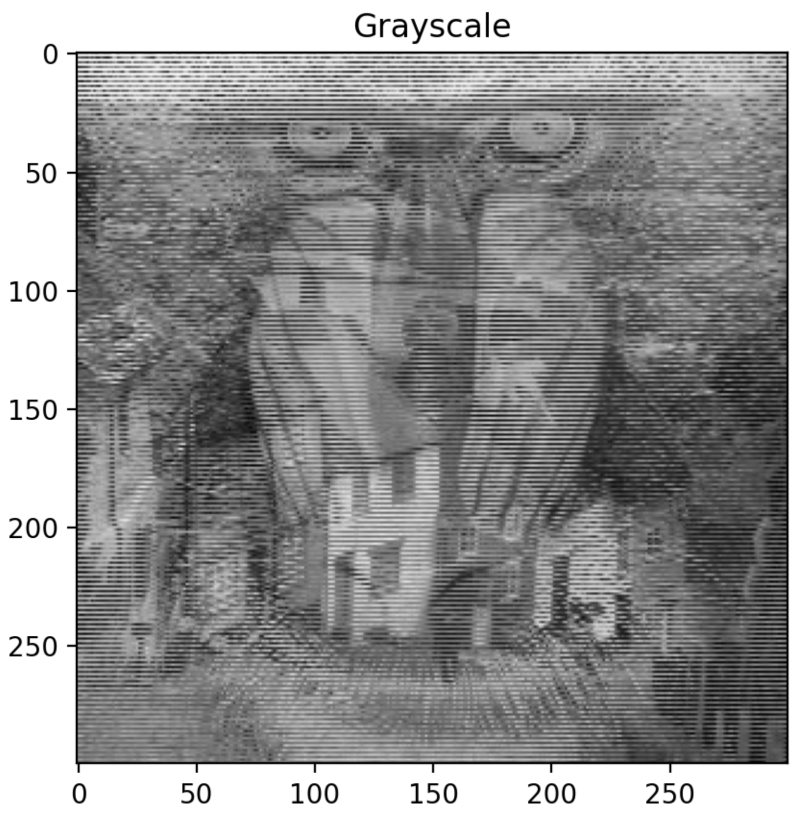
(e) Using a for loop, create a new image such that every odd numbered row is the corresponding row from image1 and the every even row is the corresponding row from image2 (Hint: Remember that indices start at 0 and not 1 in Python). Display this image in your report.
Answer Problem 3e:



(f) Accomplish the same task as part e without using a for-loop (the functions reshape and tile may be helpful here).
Answer Problem 3f:



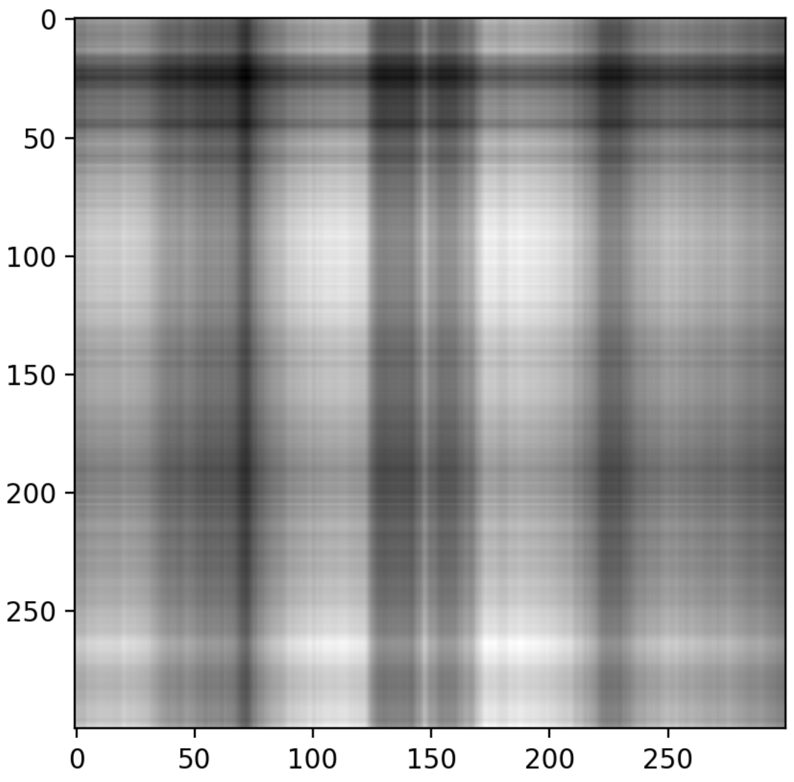
(g) Convert the result from part f to a grayscale image. Display the grayscale image with a title in your report.
Answer Problem 3g:



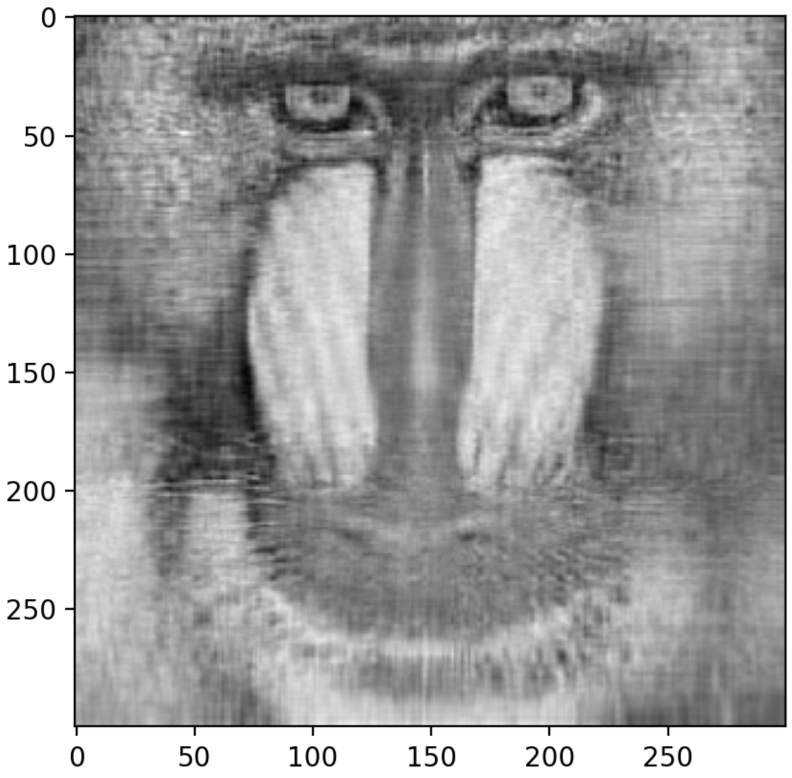
4 Singular Value Decomposition (20 points)

(a) Read in image1 as a grayscale image. Take the singular value decomposition of the image.
Answer Problem 4a:
img = io.imread('image1.jpg', as_gray=True)
u, s, v = np.linalg.svd(img, full_matrices=True)

(b) Recall from the discussion section that the best rank n approximation of a matrix is $\sum_{i=1}^{i=n} u_i \sigma_i v_i^\top$, where u_i, σ_i, and v_i are the ith left singular vector, singular value, and right singular vector respectively. Save and display the best rank 1 approximation of the (grayscale) image1 in your report.
Answer Problem 4b:



(c) Save and display the best rank 20 approximation of the (grayscale) image1 in your report.
Answer Problem 4c:


0
50
100
150
200
250
0
50
100
150
200
250