assignment1

April 3, 2020

1 Machine Learning and Computer Vision

1.1 Assigment 1

Welcome to Oversea Research Program - Machine Learning and Computer Vision. This program will give you a comprehensive introduction to computer vison providing board coverage including low level vision, inferring 3D properties from image, and object recognition. We will be using a varity of tools in this class that will require some initial configuration. To ensure everything smoothly moving forward, we will setup the majority of the tools to be used in this course in this assignment. You will also practice some basic image manipulation techniques. At the end, you will need to export this Ipython notebook as pdf.

1.1.1 Python

Python

We will use the Python programming language for all assignments in this course, with a few popular libraries (numpy, matplotlib). And assignment starters will be given in format of the browser-based Jupyter/Ipython notebook that you are currently viewing. If you have previous knowledge in Matlab, check out the numpy for Matlab users page. The section below will serve as a quick introduction on Numpy and some other libraries.

Setup Python environment

We can install Anaconda from the links given below. You can setup your environment using Anaconda for Python 2.7 or 3.6.

The Anaconda versions for Python can be downloaded from the following:

https://www.anaconda.com/download/#linux

https://www.anaconda.com/download/#macos

https://www.anaconda.com/download/#windows

After downloading and installing one of these, one needs to set the /path/to/anaconda2 in $PATH variable.

Then we can run >> jupyter notebook from terminal or use the Anaconda UI. Otherwise a more "geeky" procedure for Linux users is given here:

https://www.digitalocean.com/community/tutorials/how-to-set-up-a-jupyter-notebook-to-run-ipython-on-ubuntu-16-04.

For submitting your assignments, you can submit your python notebook file with result shown or PDF file. PDF file is needed to setup using LaTex.

Please use nbconvert tool for this. This can be installed from instructions given on: nbconvert: "conda install nbconvert" (or http://nbconvert.readthedocs.io/en/latest/install.html) The above link also gives instructions for installing Pandoc and Latex for different OS. Please follow those instructions as installing these might be required for nbconvert.

1.2 Get started with Numpy

Numpy is the fundamental package for scientific computing with Python. It provides a powerful N-dimensional array object and functions for working with these arrays.

1.2.1 Arrays

[1]:
```
import numpy as np

v = np.array([1, 0, 0])         # a 1d array
print("1d array")
print(v)
print(v.shape)                  # print the size of v
v = np.array([[1], [2], [3]])   # a 2d array
print("\n2d array")
print(v)
print(v.shape)                  # print the size of v, notice the difference
v = v.T                         # transpose of a 2d array

m = np.zeros([2, 3])            # a 2x3 array of zeros
v = np.ones([1, 3])             # a 1x3 array of ones
m = np.eye(3)                   # identity matrix
v = np.random.rand(3, 1)        # random matrix with values in [0, 1]
m = np.ones(v.shape) * 3        # create a matrix from shape
```

```
1d array
[1 0 0]
(3,)

2d array
[[1]
 [2]
 [3]]
(3, 1)
```

1.2.2 Array indexing

[2]:
```
import numpy as np

m = np.array([[1, 2, 3], [4, 5, 6]])  # create a 2d array with shape (2, 3)
print("Access a single element")
print(m[0, 2])                        # access an element
```

```
m[0, 2] = 252                          # a slice of an array is a view into the␣
 ↪same data;
print("\nModified a single element")
print(m)                               # this will modify the original array

print("\nAccess a subarray")
print(m[1, :])                         # access a row (to 1d array)
print(m[1:, :])                        # access a row (to 2d array)
print("\nTranspose a subarray")
print(m[1, :].T)                       # notice the difference of the dimension␣
 ↪of resulting array
print(m[1:, :].T)                      # this will be helpful if you want to␣
 ↪transpose it later

# Boolean array indexing
# Given a array m, create a new array with values equal to m
# if they are greater than 0, and equal to 0 if they less than or equal 0

m = np.array([[3, 5, -2], [5, -1, 0]])
n = np.zeros(m.shape)
n[m > 0] = m[m > 0]
print("\nBoolean array indexing")
print(n)
```

```
Access a single element
3

Modified a single element
[[  1   2 252]
 [  4   5   6]]

Access a subarray
[4 5 6]
[[4 5 6]]

Transpose a subarray
[4 5 6]
[[4]
 [5]
 [6]]

Boolean array indexing
[[ 3.  5.  0.]
 [ 5.  0.  0.]]
```

1.2.3 Operations on array

Elementwise Operations

[3]:
```
import numpy as np

a = np.array([[1, 2, 3], [2, 3, 4]], dtype=np.float64)
print(a * 2)                                          # scalar multiplication
print(a / 4)                                          # scalar division
print(np.round(a / 4))
print(np.power(a, 2))
print(np.log(a))


b = np.array([[5, 6, 7], [5, 7, 8]], dtype=np.float64)
print(a + b)                                          # elementwise sum
print(a - b)                                          # elementwise␣
 ↪difference
print(a * b)                                          # elementwise product
print(a / b)                                          # elementwise division
```

```
[[ 2.  4.  6.]
 [ 4.  6.  8.]]
[[ 0.25  0.5   0.75]
 [ 0.5   0.75  1.  ]]
[[ 0.  0.  1.]
 [ 0.  1.  1.]]
[[  1.   4.   9.]
 [  4.   9.  16.]]
[[ 0.          0.69314718  1.09861229]
 [ 0.69314718  1.09861229  1.38629436]]
[[  6.   8.  10.]
 [  7.  10.  12.]]
[[-4. -4. -4.]
 [-3. -4. -4.]]
[[  5.  12.  21.]
 [ 10.  21.  32.]]
[[ 0.2         0.33333333  0.42857143]
 [ 0.4         0.42857143  0.5       ]]
```

Vector Operations

[4]:
```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print("sum of array")
print(np.sum(a))                # sum of all array elements
print(np.sum(a, axis=0))        # sum of each column
print(np.sum(a, axis=1))        # sum of each row
print("\nmean of array")
```

```
print(np.mean(a))                # mean of all array elements
print(np.mean(a, axis=0))        # mean of each column
print(np.mean(a, axis=1))        # mean of each row
```

```
sum of array
10
[4 6]
[3 7]

mean of array
2.5
[ 2.  3.]
[ 1.5  3.5]
```

Matrix Operations

[5]:
```
import numpy as np

a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
print("matrix-matrix product")
print(a.dot(b))                  # matrix product
print(a.T.dot(b.T))

x = np.array([1, 2])
print("\nmatrix-vector product")
print(a.dot(x))                  # matrix / vector product
```

```
matrix-matrix product
[[19 22]
 [43 50]]
[[23 31]
 [34 46]]

matrix-vector product
[ 5 11]
```

1.2.4 SciPy image operations

SciPy builds on the Numpy array object and provides a large number of functions useful for scientific and engineering applications. We will show some examples of image operation below which are useful for this class.

[6]:
```
from imageio import imread, imsave
import numpy as np

img = imread('Lenna.png')  # read an JPEG image into a numpy array
print(img.shape)                          # print image size and color depth
```

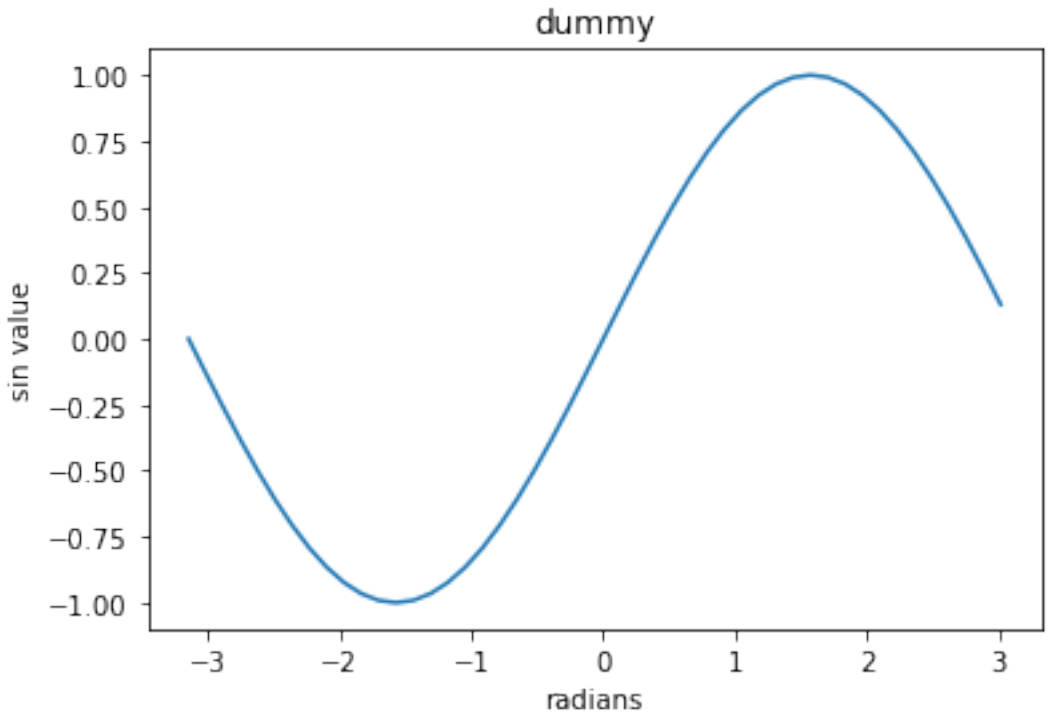
```
img_gb = img * np.array([0., 1., 1.])      # leave out the red channel
imsave('Lenna_gb.png', img_gb.astype(np.uint8))
```

```
(512, 512, 3)
```

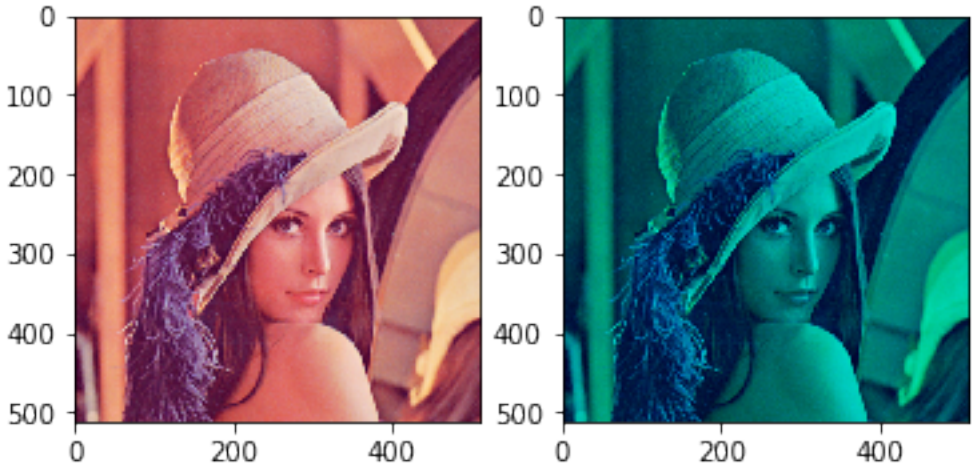
1.2.5 Matplotlib

Matplotlib is a plotting library. We will use it to show result in this assignment.

[7]:
```
# this line prepares IPython for working with matplotlib
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
import math

x = np.arange(-24, 24) / 24. * math.pi
plt.plot(x, np.sin(x))
plt.xlabel('radians')
plt.ylabel('sin value')
plt.title('dummy')

plt.show()
```



[8]:
```
# images and subplot
import numpy as np
from imageio import imread
import matplotlib.pyplot as plt

img1 = imread('Lenna.png')
img2 = imread('Lenna_gb.png')

plt.subplot(1, 2, 1)  # first plot
plt.imshow(img1)

plt.subplot(1, 2, 2) # second plot
plt.imshow(img2)
plt.show()
```



This breif overview introduces many basic functions from a few popular libraries, but is far from complete. Check out the documentations for Numpy, Scipy and Matplotlib to find out more.

1.3 Problem 1 Function

[1]:
```
# This is the most basis practices in Python.
# Please print'Welcome to Oversea Rearch Program for Computer Vision'
# to complete this problem.

import numpy as np

def fcn():
    print('Welcome to Oversea Rearch Program for Computer Vision' )
```

[2]:
```
# test the function
fcn()
```

```
Welcome to Oversea Rearch Program for Computer Vision
```

1.4 Problem 2 Matrix Manipulation

[1]:
```
import numpy as np
print('##############################################')
print('i')
A = np.array([[2, 59, 2, 5],[41, 11, 0, 4],[18, 2, 3,␣
 ↪9],[6,23,27,10],[5,8,5,1]])
B = np.array([[0,1,0,1],[0,1,1,1],[0,0,0,1],[1,1,0,1],[0,1,0,0]])
print(A, B, sep='\n')

print('##############################################')
print('ii')
C = A * B
print(C)

print('##############################################')
print('iii')
# print(C[1,:], C[4, :])
print(C[1,:]*C[4, :])

print('##############################################')
print('iv')
x, y = C.shape
max_val = np.max(C)
min_val = np.min(C)
min_loc=[]
max_loc=[]
for i in range(x):
    for j in range(y):
        if max_val == C[i,j]:
            max_loc.append((i, j))
        if min_val == C[i,j]:
            min_loc.append((i,j))
print('max = %d'%max_val)
print(max_loc)
print('min = %d'%min_val)
print(min_loc)

print('##############################################')
print('v')
x, y = C.shape
c_row = np.tile(C[0, :],(x, 1))
```

```
D = np.matrix(C - c_row)
print(D)

print('##############################################')
print('vi')
x, y = D.shape
max_val = np.max(D)
min_val = np.min(D)
min_loc=[]
max_loc=[]
for i in range(x):
    for j in range(y):
        if max_val == D[i,j]:
            max_loc.append((i, j))
        if min_val == D[i,j]:
            min_loc.append((i,j))
print('max = %d'%max_val)
print(max_loc)
print('min = %d'%min_val)
print(min_loc)
```

```
##############################################
i
[[ 2 59  2  5]
 [41 11  0  4]
 [18  2  3  9]
 [ 6 23 27 10]
 [ 5  8  5  1]]
[[0 1 0 1]
 [0 1 1 1]
 [0 0 0 1]
 [1 1 0 1]
 [0 1 0 0]]
##############################################
ii
[[ 0 59  0  5]
 [ 0 11  0  4]
 [ 0  0  0  9]
 [ 6 23  0 10]
 [ 0  8  0  0]]
##############################################
iii
[ 0 88  0  0]
##############################################
iv
max = 59
[(0, 1)]
```

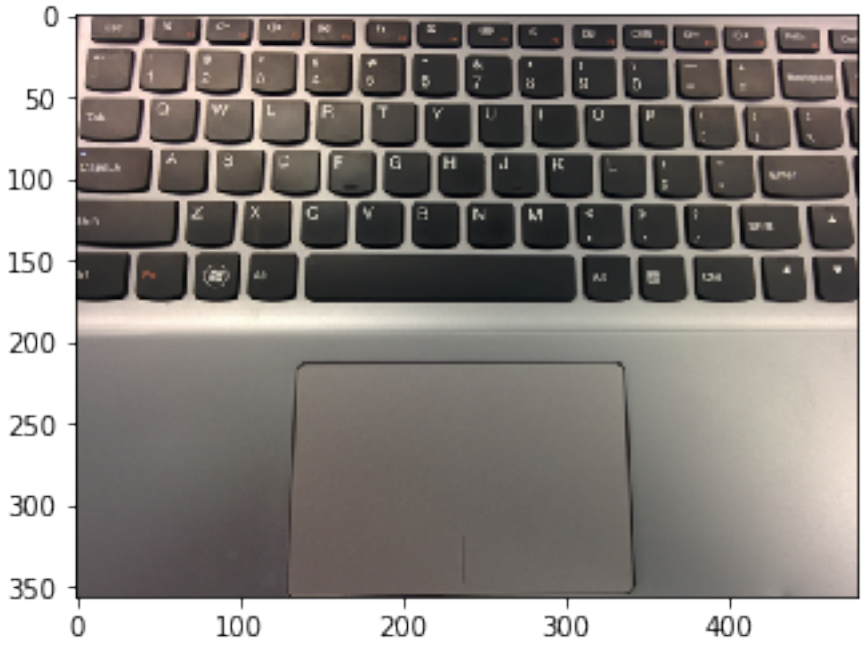
```
min = 0
[(0, 0), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1), (2, 2), (3, 2), (4, 0), (4, 2),
(4, 3)]
##############################################
v
[[  0   0   0   0]
 [  0 -48   0  -1]
 [  0 -59   0   4]
 [  6 -36   0   5]
 [  0 -51   0  -5]]
##############################################
vi
max = 6
[(3, 0)]
min = -59
[(2, 1)]
```

1.5 Problem 3 Keyboard Conundrum

In problem, you will create a function merge(img1, img2, ncols) that horizontally concatenates two perfectly aligned images. (laptop_left.png and laptop_right.png). The third argument ncols specifies the number of columns that must be deleted before the images are merged.

[35]:
```python
import numpy as np
from imageio import imread
import matplotlib.pyplot as plt

def merge(img1, img2, ncols):
    img2 = img2[:,ncols:]
    return np.hstack((img1, img2))

#Import image here
left = imread('laptop_left.png')
right = imread('laptop_right.png')

# function call
output = merge(left, right, 15)

# Plot output image
plt.imshow(output)
plt.show()
```

```
(356, 215, 3)
```



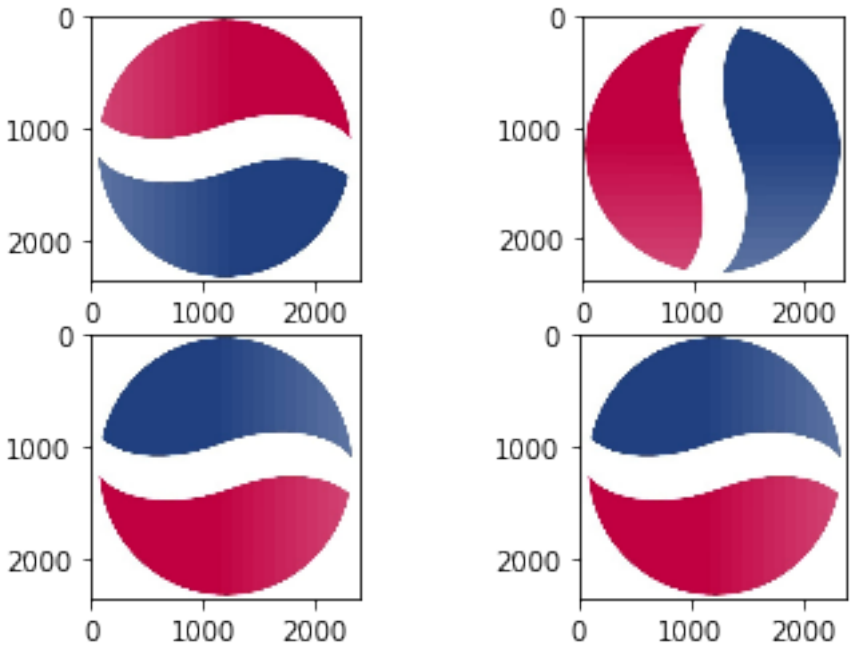
1.6 Problem 4 Image Manipulation

In the assignment folder, you will find an image "pepsi.jpg". Import this image and write your implementation for the two function signatures given below to rotate the image by 90, 180, 270 and 360 degrees anticlockwise. You must implement these functions yourself using simple array opperations (ex: numpy.rot90 and scipy.misc.imrotate are NOT allowed as they make the problem trivial). rotate and rotate90 should be out-of-place opperations (should not modify the origional image).

You should write the rest of the code to print these results in a 2X2 grid using the subplot example. The first row, first column should contain an image rotated by 90 degrees; first row, second column an image rotated by 180 degrees, second row, first column an image rotated 270 degrees and second row second column with an image rotated 360 degrees. (You may not use OpenCV function for this part.)

[158]:
```
import numpy as np
from imageio import imread
import matplotlib.pyplot as plt

# Rotate image (img) by 90 anticlockwise
def rotate90(img):
    x,y,z= img.shape
    img_res = np.full((y, x, z),255)
    for i in range(y):
        for j in range(x):
            for t in range(z):
                img_res[i, j, t] = img[j, y-i-1, t]
```

```
    return img_res

# Roate image (img) by an angle (ang) in anticlockwise direction
# Angle is assumed to be divisible by 90 but may be negative
def rotate(img, ang=0):
    assert ang%90==0
    img_res = img.copy()
    rotate_times = int(ang/90%4)
    for i in range(rotate_times):
        img_res = rotate90(img_res)
    return img_res

#Import image here
img = imread('pepsi.jpg')

#Sample call
# img90 = rotate90(img)
img90  = rotate(img, 90)
img180 = rotate(img, 180)
img360 = rotate(img, 360)

#Plotting code below
plt.subplot(2,2,1)
plt.imshow(img)
plt.subplot(2,2,2)
plt.imshow(img90)
plt.subplot(2,2,3)
plt.imshow(img180)
plt.subplot(2,2,4)
plt.imshow(img360)
plt.show()
```



1.7 Conclusion

Have you accomplished all parts of your assignment? What concepts did you used or learned in this assignment? What difficulties have you encountered? Explain your result for each section. Please wirte one or two short paragraph in the below Markdown window.

**** Your Conclusion: ****

–The concepts used to accomplished assignment 1 is basic, but it took me some time to remember it. I am familiar with python, but I don't use numpy very often. I have taken lessons on digital image processing, so these concepts are not very hard. However, in my previous DIP class, we are required to use Matlab, so image processing tools in python is new to me.

–Difficulties I have encountered: How to specify the third argument in function merge in problem 3. I inputed 15 because it showed the best merged image(I tried some numbers and selected the best one). I wonder if there is a way to program to find out the number. And I have tried to use 'equal' but it failed. If there is a way, please tell me.

The imread() output in problem 4. It was the biggest difficulties for me in assignment 1. It is understandable that it returns a 3-dimension matrix, BUT the 3 pages are not RGB pages but GB, RB, and RG??? I spent tons of time finding out I should use full 255 rather than full zeros. The output of imread() really confuses me.

–Explanations Problem 1 is easy, use print function to print it Problem 2: (i) use np.array to input matrix in numpy array object (ii) use * to do point-wise multiply operand (iii) slice row 2 and 5, then use * (iv) use 2 hier for loop to find out min and max and their indices (v) slice row 1 and tile it to make the subtraction valid (vi) repeat iv Problem 3: imread 2 images, delete the overlapping cols, hstack the two matrixs, and show it. Problem 4: function rotate90: use 2 hier for loop to put elements at places where they should be when rotate image. function rotate: calculate ang/90%4 and will know how many times it needs to rotate90. Only if ang%90==0, no matter whether it may be negative or not.

** Submission Instructions**

Remember to submit you pdf version of this notebook to Gradescope. You can find the export option at File $\rightarrow$ Download as $\rightarrow$ PDF via LaTeX