hw0

October 9, 2017

1 CSE 252A Computer Vision I Fall 2017

1.1 Assigment 0

1.2 Due Wednesday, October 11, 2017 11:59pm

Welcome to CSE252A Computer Vision I. This course 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 and submit it to Gradescope before the due date.

1.2.1 Piazza, Gradescope and Python

Piazza

Go to Piazza and sign up for the class using you ucsd.edu email account. You'll be able to ask the professor, the TAs and your classmates questions on Piazza. Class announcements will be made using Piazza, so be sure to check your email or Piazza frequently. The first problem in this assisgnment is contained in a Piazza post. Once you have logged in, please follow the instructions included in the post 'Homework 0' to complete this problem.

Gradescope

Every one of you will get a email regarding your gradescope signup once you get enrolled in this class. All the assignments are required to be submitted to gradescope for grading. Make sure that you mark each page for different problems.

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. We expect that many of you have some experience with Python and Numpy. And 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.

1.3 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.

1.3.1 Arrays

```
In [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)
```

1.3.2 Array indexing

```
In [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 da
        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 resu
        print(m[1:, :].T)                       # this will be helpful if you want to transpose

        # 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.]]
```

1.3.3 Operations on array

Elementwise Operations

```
In [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

```
In [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

```
In [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]
```

1.3.4 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.

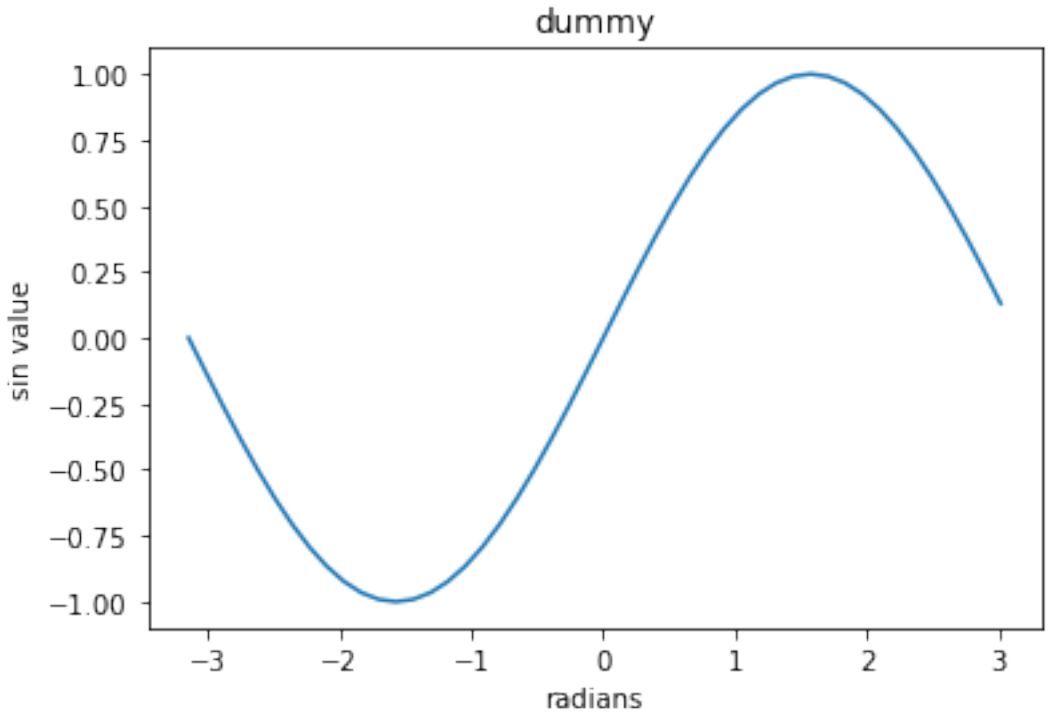
```
In [6]: from scipy.misc import imread, imsave
        import numpy as np

        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)
```

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

1.3.5 Matplotlib

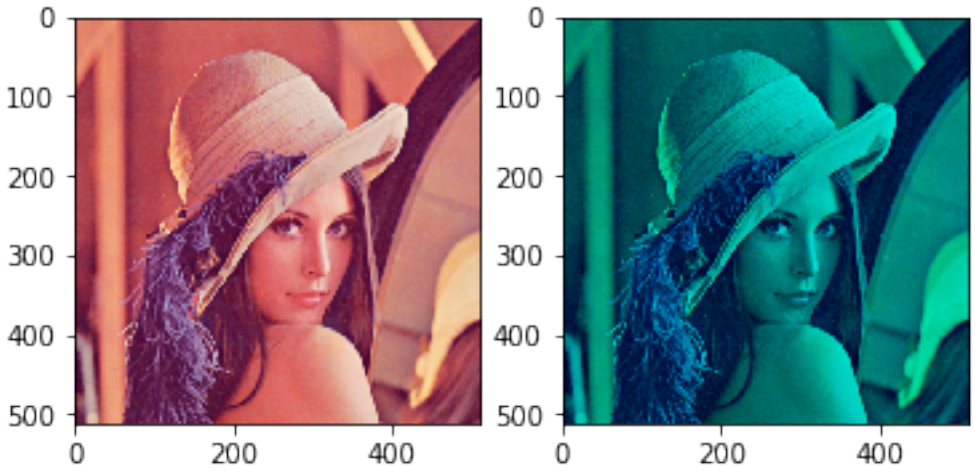
Matplotlib is a plotting library. We will use it to show result in this assignment.

```
In [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()
```




```
In [8]: # images and subplot
        import numpy as np
        from scipy.misc 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.

1.4 Problem 1 Piazza (1pt)

```
In [9]: # This problem is posted on Piazza. Sign up for the course if you have not. Then follo
        # the instructions included in the post 'Welcome to CSE252A' to complete this problem.

        import numpy as np

        def piazza():
            print 'hello world'

In [10]: # test the function
         piazza()

hello world
```

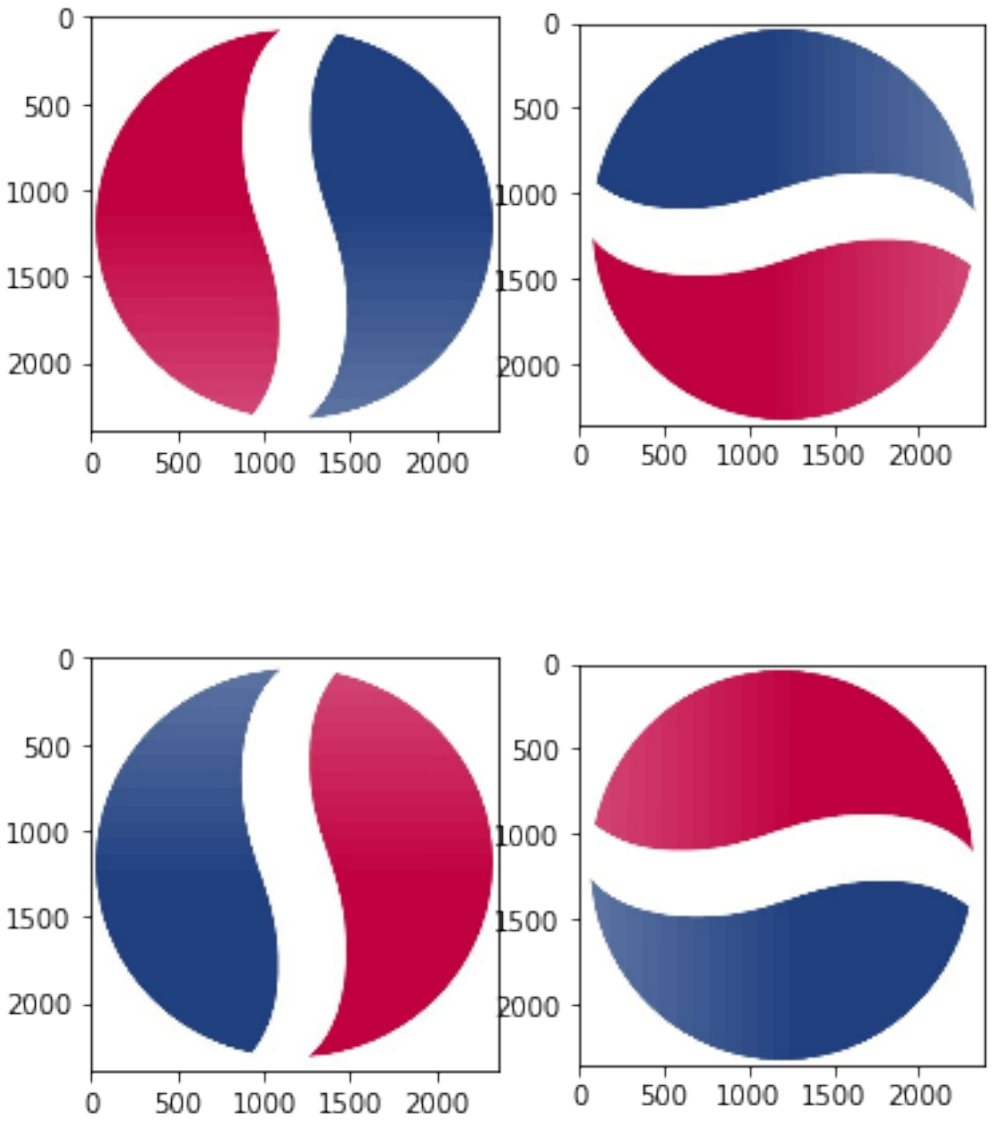
1.5 Problem 2 Image Manipulation (5pts)

In the assignment 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.

In [11]:
```
import numpy as np
from scipy.misc import imread
import matplotlib.pyplot as plt

# Rotate image (img) by 90 anticlockwise
def rotate90(img):

    m = img.shape[0];
    n = img.shape[1];
    img_res = np.zeros([n, m, 3], dtype = 'uint8');
    for i in range(3):
        img_res[:,:,i] = img[:,:,i].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

    ang%= 360;

    m = img.shape[0];
    n = img.shape[1];

    if ang==90:
        img_res = np.zeros([n, m, 3], dtype = 'uint8');
        for i in range(3):
            img_res[:,:,i] = img[:,:,i].T;
        img_res = np.flipud(img_res);

    if ang==180:
        img_res = np.flipud(img);
        img_res = np.fliplr(img_res);
```

```
    if ang==270:
        img_res = np.zeros([n, m, 3], dtype = 'uint8');
        for i in range(3):
            img_res[:,:,i] = img[:,:,i].T;
        img_res = np.fliplr(img_res);

    if ang==0:
        return img

    return img_res

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

#Sample call
img90 = rotate(img, 90)
img180 = rotate(img, 180)
img270 = rotate(img, 270)
img360 = rotate(img, 360)

#Plotting code below
plt.subplot(1,2,1)
plt.imshow(img90)

plt.subplot(1,2,2)
plt.imshow(img180)
plt.show()

plt.subplot(1,2,1)
plt.imshow(img270)

plt.subplot(1,2,2)
plt.imshow(img360)
plt.show()
```



** Submission Instructions**
Remember to submit you pdf version of this notebook to Gradescope. You can find the export option at File → Download as → PDF via LaTeX