assignment1

January 4, 2021

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

[8]:
```
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)
print(m[:2,1:3])

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
[[1 2 3]
 [4 5 6]]
[[2 3]
 [5 6]]
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.

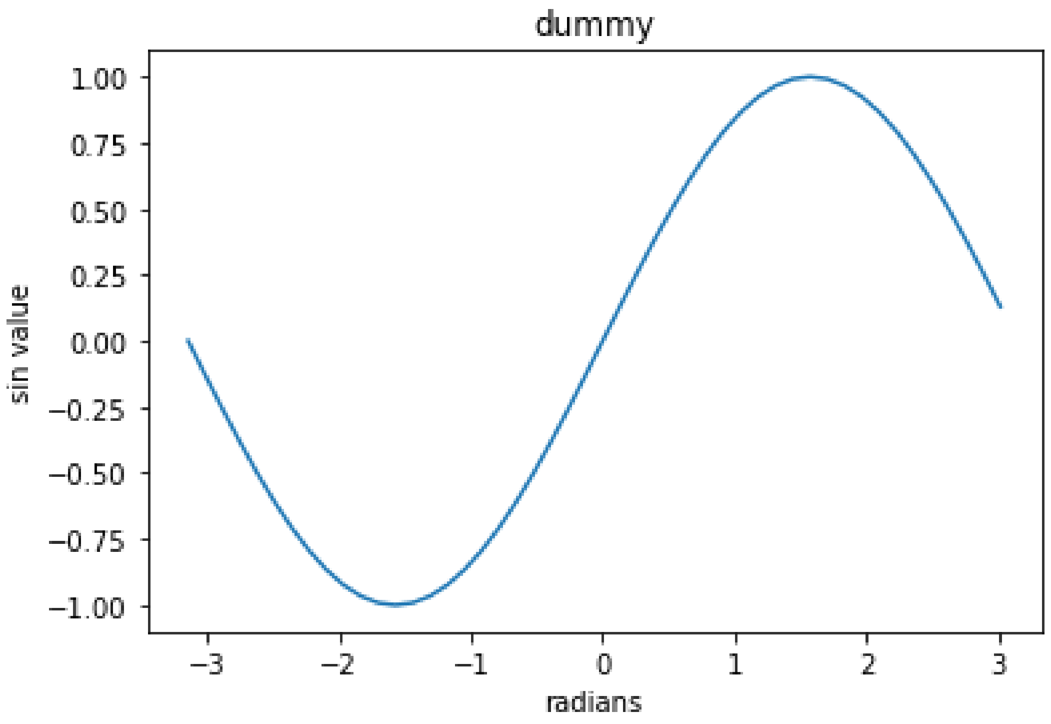
[6]:
```
from imageio import imread, imsave
import numpy as np

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

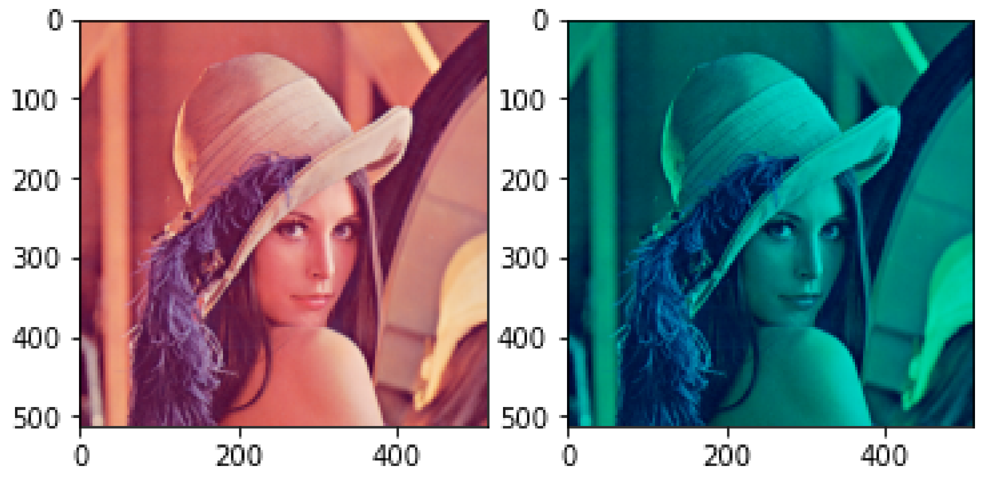
Matplotlib is a plotting library. We will use it to show result in this assignment.

[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()
```



[18]:
```python
# 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)

print(img1.shape)

plt.subplot(1, 2, 2) # second plot
plt.imshow(img2, cmap = 'gray')
plt.show()
```

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



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

[2]:
```python
# 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 Research Program for Computer Vision")
```

[3]:
```python
# test the function
fcn()
```

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

1.4 Problem 2 Matrix Manipulation

[110]:
```python
import numpy as np

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]])
C = np.multiply(A,B)
```

```
C_2nd_row_and_5th_row_inner_product = np.dot(C[1,:],C[4,:])
print(np.where(C==np.min(C)))
print(np.where(C==np.max(C)))
D = C[1:, :]
D = D[1:, :]

img = imread('peppers.png')
print(img.shape)
```

```
(array([0, 0, 1, 1, 2, 2, 2, 3, 4, 4, 4]), array([0, 2, 0, 2, 0, 1, 2, 2, 0, 2,
3]))
(array([0]), array([1]))
(384, 512)
```

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).

(Note: Print out dimension of the image, if the heigh of img1 and img2 are the same, you can directly merge two image together. When you directly merge you may see overlapped part, the overlapped part need to be removed.)

The third argument ncols specifies the number of columns that must be deleted before the images are merged.

[

Note: Image format:

Colored image has 3 layers (3D matrix), they are RGB layer, represent red, green and blue layer.

Gray Scale image and Binary image has only one layer (2D matrix).

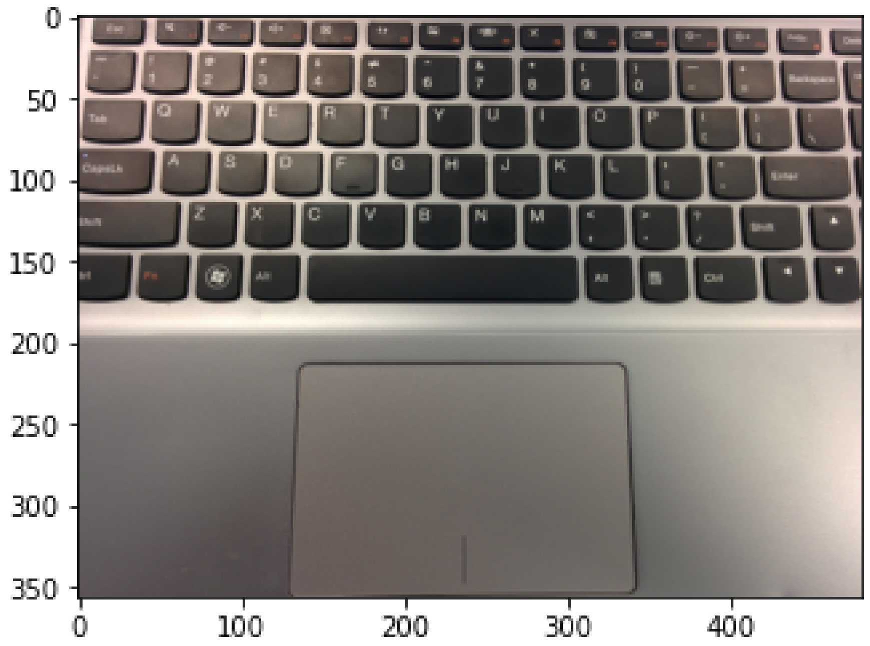
Gray Scale image has pixel intensity from 0 to 255 levels, total 256 pixel intensity.

Binary image only has black and white pixel.

]

[135]:
```
import numpy as np
from imageio import imread
import matplotlib.pyplot as plt
from scipy.sparse import coo_matrix, hstack

def merge(img1, img2):
    length_img1 = img1.shape[1]
    length_img2 = img2.shape[1]
    if(img1.shape[0] != img2.shape[0]):
        return None
    else:
        ncols = 0
        for i in range(0, length_img1):
            if ((img1[:, i] == img2[:, 0]).all()):
                ncols = length_img1 - i
        img1_without_overlap = img1[:,:-ncols]
        img = np.concatenate((img1_without_overlap,img2),axis=1)
    return img

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


# function call
# Plot output image
whole = merge(left,right)
plt.imshow(whole)
```

```
[135]: <matplotlib.image.AxesImage at 0x7f9d68226b50>
```



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).

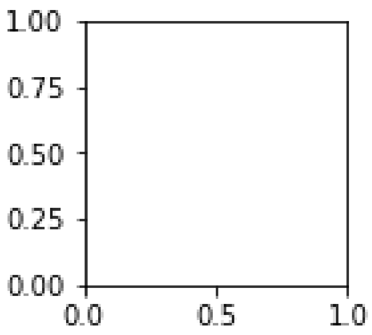
You should write the rest of the code to print these results in a 2X2 grid using the subplot example. The first row, first column should contain an image rotated by 90 degrees; first row, second column an image rotated by 180 degrees, second row, first column an image rotated 270 degrees and second row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

[98]:
```python
import numpy as np
from imageio import imread
import matplotlib.pyplot as plt

# Rotate image (img) by 90 anticlockwise help:np.transpose(), np.flipud()
def rotate90(img):
    r = img.shape[0]
    c = img.shape[1]
    if (r >= c):
        filling_part = np.zeros((r, r-c, img.shape[2]))
        img_filled = np.concatenate([img,filling_part],axis=1)
        helping_part = np.flipud(np.eye(max(img.shape)))
        img_rotate_90 =  helping_part.dot(np.transpose(img_filled))
    else:
        filling_part = np.zeros((c-r, c, img.shape[2]))
        img_filled = np.concatenate([img,filling_part],axis=0)
        helping_part = np.flipud(np.eye(max(img.shape)))
        img_rotate_90 =  helping_part.dot(np.transpose(img_filled))
    return img_rotate_90


# 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 #may need to change this line to ensure out-of-place opperation
    while ang < 0:
        ang = ang+360
    rotate_time = ang/90
    while rotate_time > 0:
        img_res = rotate90(img_res)
        rotate_time = rotate_time-1
    return img_res

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

#Sample call
img90 = rotate(img, 90)
img180 = rotate(img, 90)
```

```
img270 = rotate(img, 90)
img360 = rotate(img, 90)

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

plt.subplot(2, 2, 2) # second plot
plt.imshow(img180)

plt.subplot(2, 2, 3)  # third plot
plt.imshow(img270)

plt.subplot(2, 2, 4) # fourth plot
plt.imshow(img360)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-98-a7ce32dc0a5c> in <module>
     44 #Plotting code below
     45 plt.subplot(2, 2, 1)  # first plot
---> 46 plt.imshow(img90)
     47 
     48 plt.subplot(2, 2, 2) # second plot

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/pyplot.py in imshow(X,␣
 ↪cmap, norm, aspect, interpolation, alpha, vmin, vmax, origin, extent,␣
 ↪filternorm, filterrad, resample, url, data, **kwargs)
   2722         filternorm=True, filterrad=4.0, resample=None, url=None,
   2723         data=None, **kwargs):
-> 2724     __ret = gca().imshow(

   2725         X, cmap=cmap, norm=norm, aspect=aspect,
   2726         interpolation=interpolation, alpha=alpha, vmin=vmin,

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/__init__.py in inner(ax,␣
 ↪data, *args, **kwargs)
   1436     def inner(ax, *args, data=None, **kwargs):
   1437         if data is None:
-> 1438             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1439 
   1440         bound = new_sig.bind(ax, *args, **kwargs)

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/axes/_axes.py in␣
 ↪imshow(self, X, cmap, norm, aspect, interpolation, alpha, vmin, vmax, origin,␣
 ↪extent, filternorm, filterrad, resample, url, **kwargs)
   5521                               resample=resample, **kwargs)
   5522 
```

```
-> 5523         im.set_data(X)
   5524         im.set_alpha(alpha)
   5525         if im.get_clip_path() is None:

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/image.py in␣
 ↪set_data(self, A)
    707         if not (self._A.ndim == 2
    708                 or self._A.ndim == 3 and self._A.shape[-1] in [3, 4]):
--> 709             raise TypeError("Invalid shape {} for image data"

    710                             .format(self._A.shape))
    711

TypeError: Invalid shape (2390, 3, 2390) for image data
```



1.7 Problem 5: Sampling and Quantization

In this problem, we intend to study the effects of sampling and quantization on digital images. Your job is to write a function with the following specifications (you may use loops if necessary):

(i) The function takes one input: the image file name, ‘peppers.png’.

(ii) The input image is assumed to be grayscale.

(iii) Sample the image in spatial domain with a sampling rate of 10 (your image should be approximately 10 times smaller along width and height, do not use any numpy functions. In every 10 pixel in horizontal and vertical direction, sample 1 pixel.)

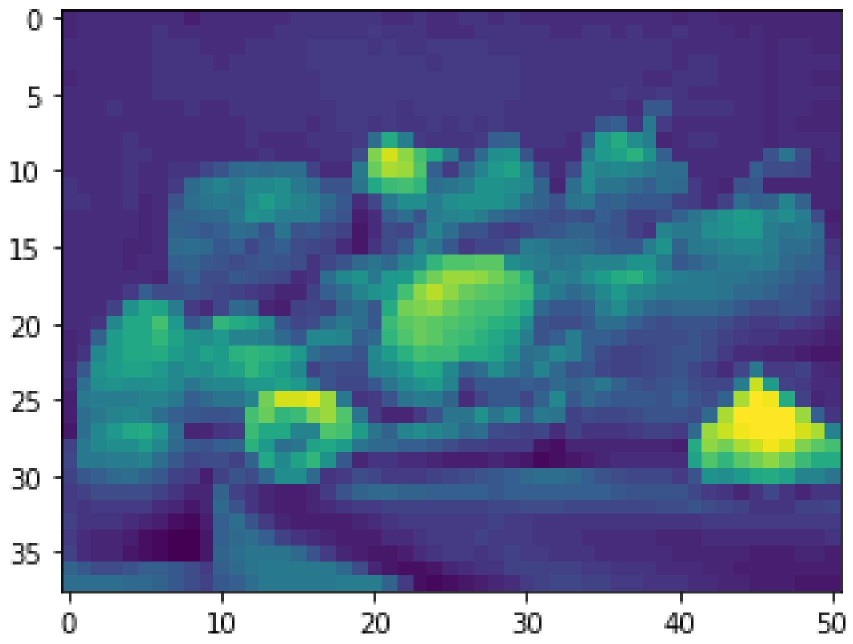
(iv) Do a 5-level uniform quantization of the sampled image so that the bins cover the whole range of grayscale values (0 to 255). You should not use any numpy functions for this.

(v) The function returns one output: the sampled and quantized image.

```
[113]: from __future__ import division   # forces floating point division
       from PIL import Image             # Python Imaging Library
       import numpy as np                # Numerical Python
       import matplotlib.pyplot as plt   # Python plotting
```

```
#Import image
img = imread('peppers.png')

def sampling_and_quantization(img):
    ratio = 10
    img_sample = np.zeros((int(img.shape[0]/ratio), int(img.shape[1]/ratio)),␣
 ↪dtype = 'float32')

    #sampling
    for i in range(img_sample.shape[0]):
        for j in range(img_sample.shape[1]):
            delta = img[i*ratio:(i+1)*ratio,j*ratio:(j+1)*ratio]
            img_sample[i,j] = np.mean(delta)

    #quantization
    level = 5
    img_quantization = img_sample
    for i in range(img_quantization.shape[0]):
        for j in range(img_quantization.shape[1]):
            img_quantization[i][j] = int(img_quantization[i][j]/level)*level
    return img_quantization

#Plotting code below
plt.imshow(sampling_and_quantization(img))
```

```
[113]: <matplotlib.image.AxesImage at 0x7f9eed9511f0>
```



1.8 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: ****

–Assignment that I haven't accomplished Problem 4

–Difficulties To find a solution to concatenate two matrixes with different width into one matrix To locate the overlapping part between two images To understand np.transpose for high-dimesional matrixes To run codes in Problem 4, because it is too time inefficient; and that is also why I can not adjust the code in short time

–Results Problem1 Printing out a sentence Problem2 Matrixes Operations Problem3 Concatenating two images Problem4 Rotating an image with a given angel (without a successful output) Problem5 Sampling and quantization of an image

---

** 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