assignment1

August 23, 2018

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

```
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)
```

1.2.2 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.]]
```

1.2.3 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]
```

1.2.4 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.

```
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)
```

1.2.5 Matplotlib

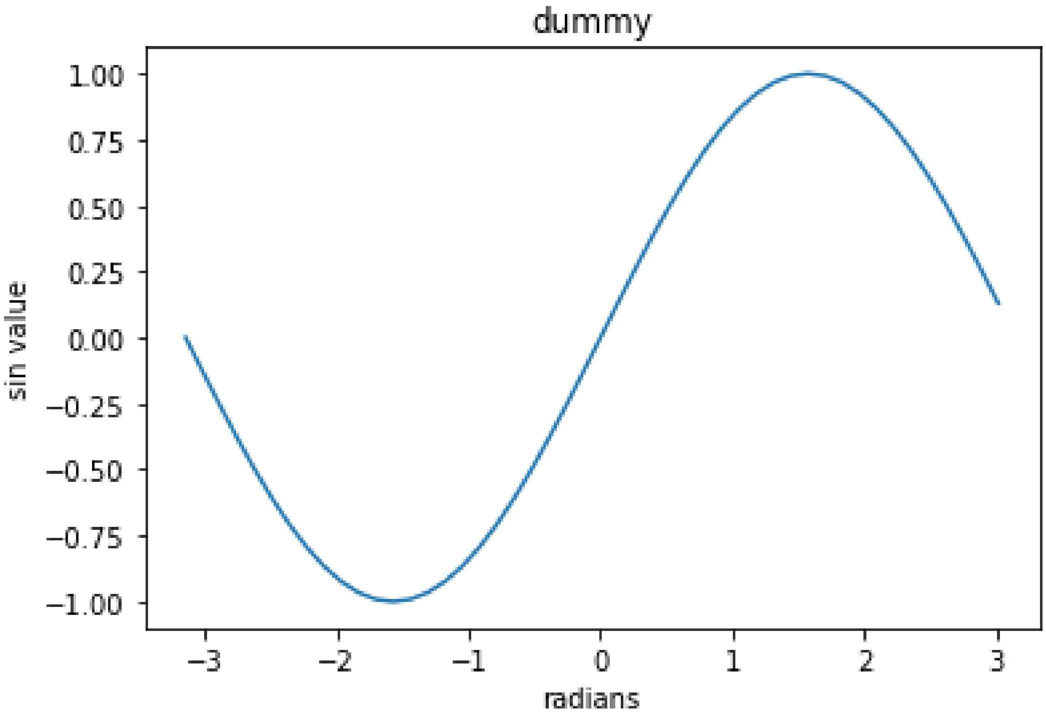
Matplotlib is a plotting library. We will use it to show result in this assignment.

```
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()
```

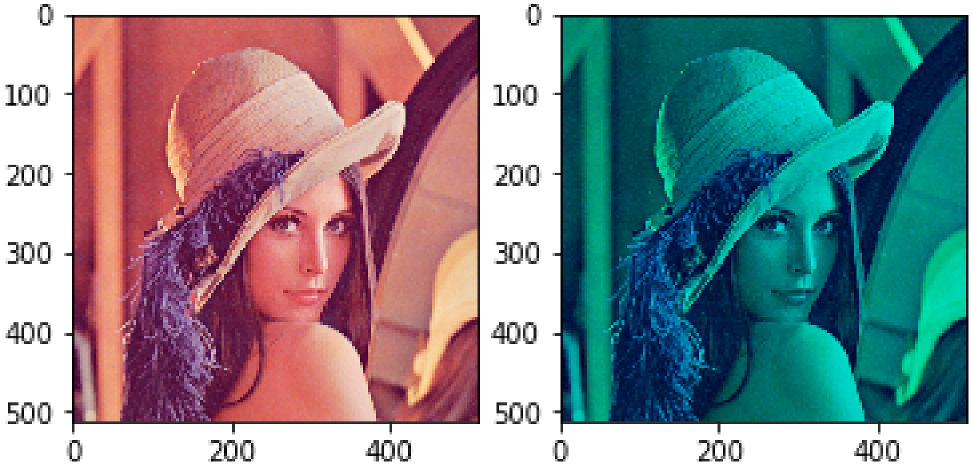


```
In [8]: # images and subplot
        import numpy as np
        from scipy.misc import imread
        import matplotlib.pyplot as plt
```

```
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.

1.3 Problem 1 Function

```
In [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")

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

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

1.4 Problem 2 Matrix Manipulation

```
In [4]: 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 = A*B
        print(C)
        print(np.sum(C[1]*C[4].T))

        # def a function for calc
        def find_num(key, array):
            if key=="max":
                row = column = 0
                t = np.max(array)
                print("max:", t)
                for r in array:
                    column = 0
                    for c in r:
                        if t == c:
                            print("row:", row+1, " column:", column+1)
                        column += 1
                    row += 1
            elif key=="min":
                row = column = 0
                t = np.min(array)
                print("min:", t)
                for r in array:
                    column = 0
                    for c in r:
                        if t == c:
                            print("row:", row+1, " column:", column+1)
                        column += 1
                    row += 1

        # invoke
        find_num("max", C)
        find_num("min", C)
        D = C - C[0]
        print(D)
        find_num("max", D)
        find_num("min", D)
```

```
[[ 0 59  0  5]
 [ 0 11  0  4]
 [ 0  0  0  9]
 [ 6 23  0 10]
 [ 0  8  0  0]]
88
max: 59
row: 1  column: 2
min: 0
row: 1  column: 1
row: 1  column: 3
row: 2  column: 1
row: 2  column: 3
row: 3  column: 1
row: 3  column: 2
row: 3  column: 3
row: 4  column: 3
row: 5  column: 1
row: 5  column: 3
row: 5  column: 4
[[  0   0   0   0]
 [  0 -48   0  -1]
 [  0 -59   0   4]
 [  6 -36   0   5]
 [  0 -51   0  -5]]
max: 6
row: 4  column: 1
min: -59
row: 3  column: 2
```

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.

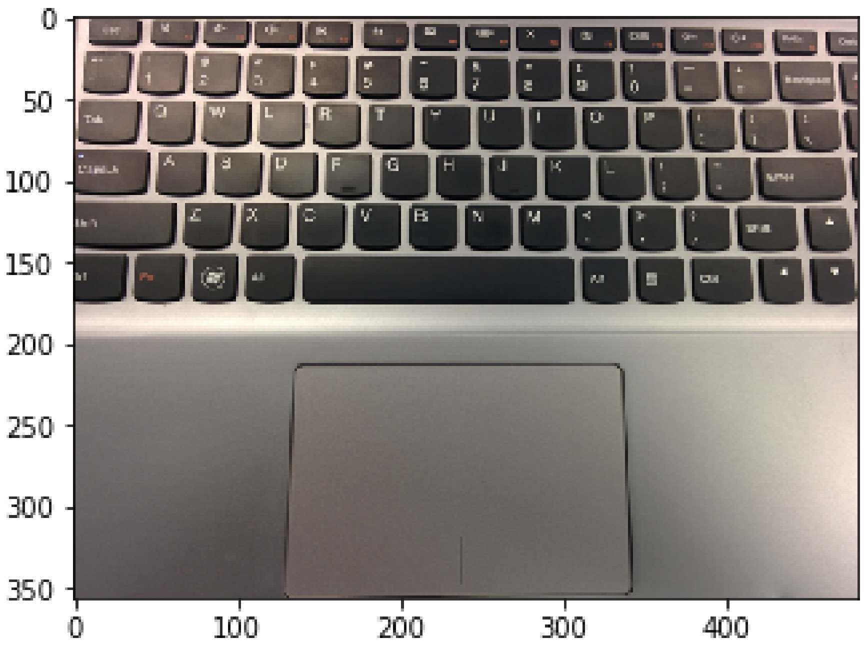
```
In [6]: import numpy as np
        from scipy.misc import imread
        import matplotlib.pyplot as plt


        def merge(i1, i2, column):
            i2 = i2[:, column:, :]
```

```
    img = np.hstack((i1, i2))
    plt.subplot(1, 1, 1)  # the only plot
    plt.imshow(img)
    plt.show()
```

```
img1 = imread('laptop_left.png')
img2 = imread('laptop_right.png')
# try by myself to find out the number of ncols
merge(img1, img2, 14)
```

```
C:\study\anaconda\lib\site-packages\ipykernel_launcher.py:15: DeprecationWarning: `imread` is d
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
  from ipykernel import kernelapp as app
C:\study\anaconda\lib\site-packages\ipykernel_launcher.py:16: DeprecationWarning: `imread` is d
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
  app.launch_new_instance()
```



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.)

```
In [7]: import numpy as np
        from scipy.misc import imread
        import matplotlib.pyplot as plt


        # Rotate image (img) by 90 anticlockwise
        def rotate90(array):
            sp = array.shape
            tem = np.array([[[0, 0, 0]] * sp[0]] * sp[1])
            for x in range(sp[0]):
                for y in range(sp[1]):
                    tem[sp[1] - y - 1][x] = array[x][y]
            return tem


        # 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 = ang % 360
            while ang < 0:
                ang += 90
            while ang != 0:
                img = rotate90(img)
                ang -= 90
            return img


        # Import image here
        img1 = imread('pepsi.jpg')

        # Sample call
        img90 = rotate(img1, 90)
        img180 = rotate(img1, 180)
        img270 = rotate(img1, 270)
        img360 = rotate(img1, 360)
        # 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)
plt.show()
```

```
C:\study\anaconda\lib\site-packages\ipykernel_launcher.py:30: DeprecationWarning: `imread` is
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

C:\study\anaconda\lib\site-packages\IPython\core\formatters.py in __call__(self, obj)
339                 pass
340             else:
--> 341                 return printer(obj)
342             # Finally look for special method names
343             method = get_real_method(obj, self.print_method)


C:\study\anaconda\lib\site-packages\IPython\core\pylabtools.py in <lambda>(fig)
236
237     if 'png' in formats:
--> 238         png_formatter.for_type(Figure, lambda fig: print_figure(fig, 'png', **kwar
239     if 'retina' in formats or 'png2x' in formats:
240         png_formatter.for_type(Figure, lambda fig: retina_figure(fig, **kwargs))


C:\study\anaconda\lib\site-packages\IPython\core\pylabtools.py in print_figure(fig, fm
120
121     bytes_io = BytesIO()
--> 122     fig.canvas.print_figure(bytes_io, **kw)
123     data = bytes_io.getvalue()
124     if fmt == 'svg':


C:\study\anaconda\lib\site-packages\matplotlib\backend_bases.py in print_figure(self,
   2214                     orientation=orientation,
   2215                     dryrun=True,
-> 2216                     **kwargs)
```

```
   2217                 renderer = self.figure._cachedRenderer
   2218                 bbox_inches = self.figure.get_tightbbox(renderer)


    C:\study\anaconda\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(sel
    505 
    506     def print_png(self, filename_or_obj, *args, **kwargs):
--> 507         FigureCanvasAgg.draw(self)
    508         renderer = self.get_renderer()
    509         original_dpi = renderer.dpi


    C:\study\anaconda\lib\site-packages\matplotlib\backends\backend_agg.py in draw(self)
    428             # if toolbar:
    429             #     toolbar.set_cursor(cursors.WAIT)
--> 430             self.figure.draw(self.renderer)
    431         finally:
    432             # if toolbar:


    C:\study\anaconda\lib\site-packages\matplotlib\artist.py in draw_wrapper(artist, rende
     53                 renderer.start_filter()
     54 
---> 55             return draw(artist, renderer, *args, **kwargs)
     56         finally:
     57             if artist.get_agg_filter() is not None:


    C:\study\anaconda\lib\site-packages\matplotlib\figure.py in draw(self, renderer)
   1297 
   1298             mimage._draw_list_compositing_images(
-> 1299                 renderer, self, artists, self.suppressComposite)
   1300 
   1301             renderer.close_group('figure')


    C:\study\anaconda\lib\site-packages\matplotlib\image.py in _draw_list_compositing_image
    136     if not_composite or not has_images:
    137         for a in artists:
--> 138             a.draw(renderer)
    139     else:
    140         # Composite any adjacent images together


    C:\study\anaconda\lib\site-packages\matplotlib\artist.py in draw_wrapper(artist, rende
     53                 renderer.start_filter()
     54 
---> 55             return draw(artist, renderer, *args, **kwargs)
```

```
     56         finally:
     57             if artist.get_agg_filter() is not None:


    C:\study\anaconda\lib\site-packages\matplotlib\axes\_base.py in draw(self, renderer, i
   2435             renderer.stop_rasterizing()
   2436
-> 2437         mimage._draw_list_compositing_images(renderer, self, artists)
   2438
   2439         renderer.close_group('axes')


    C:\study\anaconda\lib\site-packages\matplotlib\image.py in _draw_list_compositing_imag
    136     if not_composite or not has_images:
    137         for a in artists:
--> 138             a.draw(renderer)
    139     else:
    140         # Composite any adjacent images together


    C:\study\anaconda\lib\site-packages\matplotlib\artist.py in draw_wrapper(artist, rende
     53                 renderer.start_filter()
     54
---> 55             return draw(artist, renderer, *args, **kwargs)
     56         finally:
     57             if artist.get_agg_filter() is not None:


    C:\study\anaconda\lib\site-packages\matplotlib\image.py in draw(self, renderer, *args,
    564         else:
    565             im, l, b, trans = self.make_image(
--> 566                 renderer, renderer.get_image_magnification())
    567             if im is not None:
    568                 renderer.draw_image(gc, l, b, im)


    C:\study\anaconda\lib\site-packages\matplotlib\image.py in make_image(self, renderer,
    791         return self._make_image(
    792             self._A, bbox, transformed_bbox, self.axes.bbox, magnification,
--> 793             unsampled=unsampled)
    794
    795     def _check_unsampled_image(self, renderer):


    C:\study\anaconda\lib\site-packages\matplotlib\image.py in _make_image(self, A, in_bbo
    477                     A, output, t, _interpd_[self.get_interpolation()],
    478                     self.get_resample(), alpha,
--> 479                     self.get_filternorm() or 0.0, self.get_filterrad() or 0.0)
```

```
    480
    481             # at this point output is either a 2D array of normed data


    ValueError: 3-dimensional arrays must be of dtype unsigned byte, unsigned short, float3
```

```
<matplotlib.figure.Figure at 0x29b5c2b8a58>
```

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: ****

--Thanks to the first lesson, I reviewed some important things in python. In the problem 4, I tried to use some more effective ways to do rotating but failed. So I run this code in pycharm for a few minutes, and finally get the image.(wonder why can't show it in ipython :( ) So i have to show it as attachment.

** 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