Lecture 8 Matplotlib and Image Processing

Matplotlib

Matplotlib (https://matplotlib.org/) serves as the package to produce publication-quality figures in Python, and provides interface closely resembling to matlab (https://matplotlib.org/tutorials/introductory/pyplot.html).

In [ ]:
```
import matplotlib as mpl # import whole package
import matplotlib.pyplot as plt # or just import submodule pylot, providing matlab-li
ke functions
# these are "standard shorthands", though some poeple use other nicknames
```

In [ ]:
```
dir(mpl)
```

In [ ]:
```
dir(plt)
```

Of course you can explore the Github (https://github.com/matplotlib/matplotlib/tree/master/lib/matplotlib) to see the source codes if you like.

In [ ]:
```
help(plt.plot)
```

Basic usage of pyplot: Very similiar to Matlab

In [ ]:
```
import numpy as np
x = np.linspace(0, 10, 100)
fig = plt.figure(figsize=(8, 6)) # create the figure, just like figure() in matlab
plt.plot(x, np.sin(x), linestyle = '-',color = 'b',label='sin')  # label is used for
 legend
plt.plot(x, np.cos(x), '--g', label = 'cos')
plt.xlim(-1, 11)
plt.title("A Sine Curve")
plt.xlabel('x')
plt.ylabel("sin(x)")
plt.legend()
```

Of course there is some object-oriented feature.

In [ ]:
```
type(fig)
```

In [ ]:
```
dir(fig)
```

In [ ]:
```
fig.savefig('myfigure.png') # savefig is just a method of instance fig!
```

The object-oriented feature is more evident in making subplots. Explore more usages here (https://matplotlib.org/3.1.0/gallery/subplots_axes_and_figures/subplots_demo.html).

In [ ]:
```
# subplots
fig, ax = plt.subplots(2, dpi = 100)
ax[0].plot(x, np.sin(x)) # plot and set_title are the methods of ax[0] -axes
ax[0].set_title('sin')
ax[1].plot(x, np.cos(x))
ax[1].set_title('cos')
```

Distinguish the concept of axes and axis in Matplotlib (https://matplotlib.org/faq/usage_faq.html)

In [ ]:
```
type(ax)
```

In [ ]:
```
type(ax[0])
```

In [ ]:
```
fig
```

Image Processing

There are many great packages available to handle the image data in Python, such as Pillow (https://pillow.readthedocs.io/en/stable/handbook/tutorial.html#using-the-image-class), Scikit-Image (https://scikit-image.org/) and opencv-python (https://github.com/skvark/opencv-python).

Here we import images from Scikit-Image which is well-compatible with Numpy (https://scikit-image.org/docs/dev/user_guide/numpy_images.html), and use Numpy to manipulate images.

In [ ]:
```
from skimage import data
image_astro = data.astronaut()# read the image as numpy array
image_rock = data.rocket()
fig = plt.figure(dpi=100)
plt.imshow(image_astro)
```

In [ ]:
```
fig = plt.figure(dpi=100)
plt.imshow(image_rock)
```

In [ ]:
```
image_astro.shape
```

In [ ]:
```
image_rock.shape
```

In [ ]:
```
[np.max(image_astro),np.min(image_astro)]
```

Even with simple Numpy expressions, you can do some image processing like in Photoshop!

- Crop the images

In [ ]:
```
image_astro_split = image_astro[:427,:,:]
image_rock_split = image_rock[:,:512,:]
```

In [ ]:
```
fig, ax = plt.subplots(ncols=2, dpi = 100)
ax[0].imshow(image_astro_split) # plot and set_title are the methods of ax[0] -axes
ax[1].imshow(image_rock_split)
```

- Invert the colors

In [ ]:
```
fig = plt.figure(dpi=100)
plt.imshow(255-image_rock)
```

- Exchange RGB channels

In [ ]:
```
fig = plt.figure(dpi=100)
plt.imshow(image_rock[:,:,[2,1,0]])
```

- Binarize the image

In [ ]:
```
image = image_rock
image_bi = np.empty_like(image)

thresh = 90
maxval = 255

for i in range(3):
    image_bi[:, :, i] = (image[:, :, i] > thresh) * maxval

fig = plt.figure(dpi=100)
plt.imshow(image_bi[:,:,[2,1,0]])
```

In [ ]:
```
image_bi
```

- Blending

In [ ]:
```
image_combine = 0.4*image_astro_split+0.6*image_rock_split
fig = plt.figure(dpi=100)
plt.imshow(image_combine.astype('uint8'))
plt.axis('off')
```