Lab 4: Getting started with image processing using scikit-image

In this lab you will load and process image data to perform some basic tasks and visualize the results.

Task 0: Software setup

Install the following library for image processing.

- Scikit-Image: Open-source machine learning library for image processing. Ref: https://scikit-image.org/docs/stable/install.html

Task 1: Load and visualize an image

- Create a new python file either by clicking in the file tab, file-> New-> Python file or by right clicking in the folder right click -> New -> Python file
- Let's begin with the basics and use the following code to load an image.
- Visualize the provided images: 'coins.jpg' and 'astronaut.jpg'.

```python
import skimage
import os
import matplotlib.pyplot as plt

filename_path = '/your directory/'
filename = os.path.join(filename_path,'coins.jpg')

from skimage import io

image = io.imread(filename)
```

- Check the size of the image with ".shape", the type of the image with ".dtype" and access the intensity level for a random pixel of the image with "[1,100,1]". Use the following code and observe this information for the "coins" and "astronaut" images.
- Visualize the image with either the "io.imshow" or the "plt.imshow" commands.

```python
print(image.shape)
print(image.dtype)
print(image[1,100,1])
```

```python
io.imshow(image)
io.show()
```

```python
ax = plt.imshow(image)
plt.show()
```

Task 2: Color space conversion

- Next step is to work with color space conversion. In this task you convert a color image to a grayscale image.
- Use the following sample code.

```
import skimage
from skimage.color import rgb2gray
import os

filename_path = '/your directory/'
filename = os.path.join(filename_path,'coins.jpg')

from skimage import io


image = io.imread(filename)
grayscale = rgb2gray(image)
```

- Convert the “coins” and “astronaut” images to grayscale.
- Explain what happens with the dtype and size of each image before and after the conversion.

Task 3: Image rescale and resize

- In this part you will modify the size and scale of images.
- Scikit-image offers the image rescale function for using a factor and the resize function to change the size of the image with the desired width and length.
- Use the following sample code.

```
import skimage
from skimage.color import rgb2gray
from skimage.transform import rescale, resize
import os

filename_path = '/your directory/'
filename = os.path.join(filename_path,'coins.jpg')

from skimage import io

image = io.imread(filename)
grayscale = rgb2gray(image)
image_rescaled = rescale(grayscale, 0.5, anti_aliasing=False)


image_resized=resize(grayscale, (image.shape[0] // 4, image.shape[1]// 4),
anti_aliasing=True)
```

- Resize the “astronaut” image to any desired width and height value of your choice.
- Rescale the “astronaut” image by factors of 0.75, 0.5 and 0.25.
- Observe the differences in the operations.

Task 4: Image thresholding

- In this part you will segment an image with the basic thresholding operation.
- Use the following sample code.

```python
import skimage
import os
import numpy as np
from skimage.color import rgb2gray
from skimage.exposure import histogram
import matplotlib.pyplot as plt
from skimage.util import img_as_ubyte

filename_path = '/your directory/'
filename = os.path.join(filename_path,'coins.jpg')

from skimage import io

image = io.imread(filename)
grayscale = rgb2gray(image)

ax = plt.hist(grayscale.ravel(), bins = 256)

t = ??
binary = grayscale < t

fig, ax = plt.subplots()
plt.imshow(binary, cmap="gray")
plt.show()
```

Use the grayscale converted “coins” image from Task 2, calculate the histogram and select a threshold to distinguish the object from the background. Based on the histogram, select a value for the variable t and replace the ?? on line 19 with your selected value. Observe the data type of the grayscale image and select the t value accordingly (e.g uint8 or float64).

Task 5: Template matching

- In this task you apply the template matching algorithm.
- Use the following sample code and explain the result.

```python
import numpy as np
import matplotlib.pyplot as plt

from skimage import data
from skimage.feature import match_template


image = data.coins()
coin = image[170:220, 75:130]

result = match_template(image, coin)
ij = np.unravel_index(np.argmax(result), result.shape)
x, y = ij[::-1]

fig = plt.figure(figsize=(8, 3))
ax1 = plt.subplot(1, 3, 1)
ax2 = plt.subplot(1, 3, 2)
ax3 = plt.subplot(1, 3, 3, sharex=ax2, sharey=ax2)

ax1.imshow(coin, cmap=plt.cm.gray)
ax1.set_axis_off()
ax1.set_title('template')

ax2.imshow(image, cmap=plt.cm.gray)
ax2.set_axis_off()
ax2.set_title('image')
# highlight matched region
hcoin, wcoin = coin.shape
rect = plt.Rectangle((x, y), wcoin, hcoin, edgecolor='r', facecolor='none')
ax2.add_patch(rect)

ax3.imshow(result)
ax3.set_axis_off()
ax3.set_title('`match_template`\nresult')
# highlight matched region
ax3.autoscale(False)
ax3.plot(x, y, 'o', markeredgecolor='r', markerfacecolor='none', markersize=10)

plt.show()
```

Task 6: Own implementation of template matching algorithm

- In this task you use the coin data from Task 5 and implement your own version of template matching.
- Use your preferred reference to implement your own code, such as the following:

https://www.ijcaonline.org/archives/volume153/number10/swaroop-2016-ijca-912165.pdf