

Single-board Computer - Raspberry-Pi & Programming

Dr. Soharab Hossain Shaikh
Associate Professor, CSE, SoET, BMU

1

Plan for the upcoming 4 Sessions

Monday, 20th September (completed last week)

- Setting the Context (Why RaspberryPi?)
- Introduction, Installation & Setup - RaspberryPi

Tuesday, 21st September (completed last week)

- A Quick Intro to Python Programming
- Interfacing Sensors to RaspberryPi and GPIO Programming

Monday, 27th September (today)

- Image Processing with Python - OpenCV
- Exploring new tools/libraries

Tuesday, 28th September (next week)

- Build Web-interface with Python - Flask
- Run a full-fledged applications with Raspberry-Pi

2



Session - 3

- **Image Processing with Python & OpenCV**
 - **Exploring new Python Libraries**

3

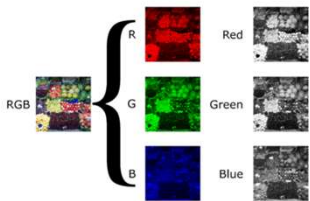
Image Processing



https://docs.opencv.org/master/d9/df8/tutorial_root.html

4

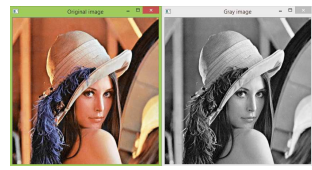
RGB Image



OpenCV
Open Source Computer Vision

5

RGB to Gray



```
import cv2
image = cv2.imread('C:/Users/ll/Desktop/Test.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Original image', image)
cv2.imshow('Gray image', gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Grayscale = $(R + G + B) / 3$.


$$Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

OpenCV
Open Source Computer Vision

6

Image Blurring



```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread('opencv-logo-white.png')
blur = cv.GaussianBlur(img, (5,5), 0)

plt.subplot(121), plt.imshow(img), plt.title('Original')
plt.xticks([], plt.yticks([]))
plt.subplot(122), plt.imshow(blur), plt.title('Blurred')
plt.xticks([], plt.yticks([]))
plt.show()
```

https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

OpenCV
Open Source Computer Vision

7

Edge Detection



```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('messi1.jpg')
edges = cv.Canny(img, 100, 200)

plt.subplot(121), plt.imshow(img, cmap = 'gray')
plt.title('Original Image'), plt.xticks([], plt.yticks([]))
plt.subplot(122), plt.imshow(edges, cmap = 'gray')
plt.title('Edge Image'), plt.xticks([], plt.yticks([]))
plt.show()
```

https://docs.opencv.org/master/da/d22/tutorial_py_canny.html

OpenCV
Open Source Computer Vision

8

Exploring new Libraries/Tools

9



Scikit Image
image processing in python

Installation Gallery Documentation Community Source Search documentation...

Stable (release notes)
0.18.3 - August 2021
Download

Development
pre-0.19
Download

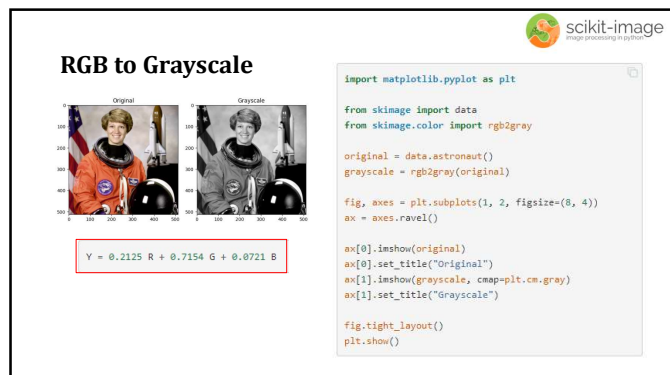
Image processing in Python
Scikit image is a collection of algorithms for image processing. It is available free of charge and free of restriction. We pride ourselves on high-quality, peer-reviewed code, written by an active community of volunteers.
@cristian

If you find this project useful, please cite: [BIBTeX]
Stefan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulgogue, Joshua D. Warner, Neil Yager, Emmanuelle Goullart, Tony Yu and the scikit-image contributors. **scikit-image: Image processing in Python**. PeerJ 2:e453 (2014) <https://doi.org/10.7717/peerj.453>

<https://scikit-image.org/>

GitHub source & bug reports
Contribute get involved
Mailing List dev discussion
Forum advice & community
StackOverflow code help

10



RGB to Grayscale

Original Grayscale

$Y = 0.2125 R + 0.7154 G + 0.0721 B$

```
import matplotlib.pyplot as plt

from skimage import data
from skimage.color import rgb2gray

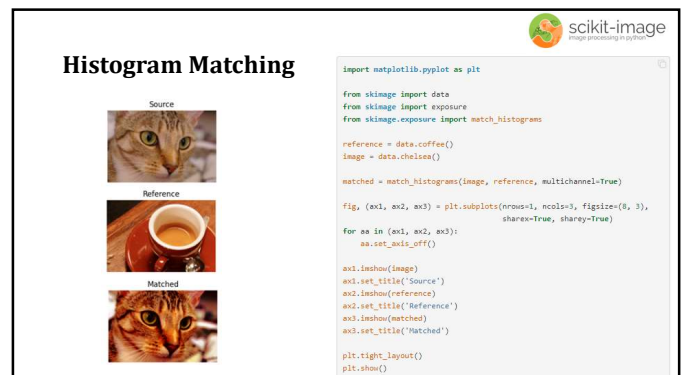
original = data.astronaut()
grayscale = rgb2gray(original)

fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()

ax[0].imshow(original)
ax[0].set_title("Original")
ax[1].imshow(grayscale, cmap=plt.cm.gray)
ax[1].set_title("Grayscale")

fig.tight_layout()
plt.show()
```

11



Histogram Matching

Source Reference Matched

```
import matplotlib.pyplot as plt

from skimage import data
from skimage import exposure
from skimage.exposure import match_histograms

reference = data.coffee()
image = data.chelsea()

matched = match_histograms(image, reference, multichannel=True)


fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(8, 3),
                                   sharex=True, sharey=True)
for aa in (ax1, ax2, ax3):
    aa.set_axis_off()

ax1.imshow(image)
ax1.set_title('Source')
ax2.imshow(reference)
ax2.set_title('Reference')
ax3.imshow(matched)
ax3.set_title('Matched')

plt.tight_layout()
plt.show()
```

12

Edge Detection



noisy image

Canny filter, $\sigma=1$

Canny filter, $\sigma=3$

```

import numpy as np
import matplotlib.pyplot as plt
from skimage.feature import feature

# Generate noisy image of a square
(x, y) = np.meshgrid(256, 256)
x2 = (x - 128) ** 2 + (y - 128) ** 2
x2 = np.sqrt(x2)
x2 = np.clip(x2, 0, 1)
x2 = 0.5 * x2
x2 = np.random.uniform(0, 1, x2.shape)
x2 = x2 * x2

# Compute the Canny filter for two values of sigma
edges1 = feature.canny(x2, sigma=1)
edges2 = feature.canny(x2, sigma=3)

# Display results
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(8, 4), sharex=True, sharey=True)

ax1.imshow(x2, cmap=plt.cm.gray)
ax1.set_title('noisy image', fontsize=10)

ax2.imshow(edges1, cmap=plt.cm.gray)
ax2.set_title('Canny filter, sigma=1', fontsize=10)


ax3.imshow(edges2, cmap=plt.cm.gray)
ax3.set_title('Canny filter, sigma=3', fontsize=10)

plt.tight_layout()
plt.show()

```

13

HoG Feature Descriptor



Input image

Histogram of Oriented Gradients

```

import matplotlib.pyplot as plt

from skimage.feature import hog
from skimage import data, exposure

image = data.astronaut()

fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),
                    cells_per_block=(1, 1), visualize=True, multichannel=True)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4), sharex=True, sharey=True)

ax1.axis('off')
ax1.imshow(image, cmap=plt.cm.gray)
ax1.set_title('Input image')

# Rescale histogram for better display
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

ax2.axis('off')
ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
ax2.set_title('Histogram of Oriented Gradients')
plt.show()

```

14

NumPy


What is NumPy?

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices) and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebraic operations, random simulations and much more.

```

>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int64'
>>> a.itemsize
8
>>> a.size
15
>>> type(a)
<class 'numpy.ndarray'>
>>> b = np.array([6, 7, 8])
>>> type(b)
<class 'numpy.ndarray'>

```



NumPy

NumPy is a community-driven project. Development happens on GitHub. NumPy is fiscally sponsored by NumFOCUS.

<https://www.numpy.org/>

https://numpy.org/doc/stable/user/tutorials_index.html

15

SciPy

SciPy.org

SciPy library

The SciPy library is one of the core packages that make up the SciPy stack. It provides many user-friendly and efficient numerical routines, such as routines for numerical integration, interpolation, linear algebra, and statistics. For tutorials, reference documentation, the SciPy roadmap, and a contributor guide, please see the documentation. SciPy is a community-driven project. Development happens on GitHub. SciPy is fiscally sponsored by NumFOCUS.

SciPy library

- Library (library)
- License
- Code of Conduct
- Documentation
- Build instructions
- Reporting bugs
- Developer zone
- Donations
- FAQ

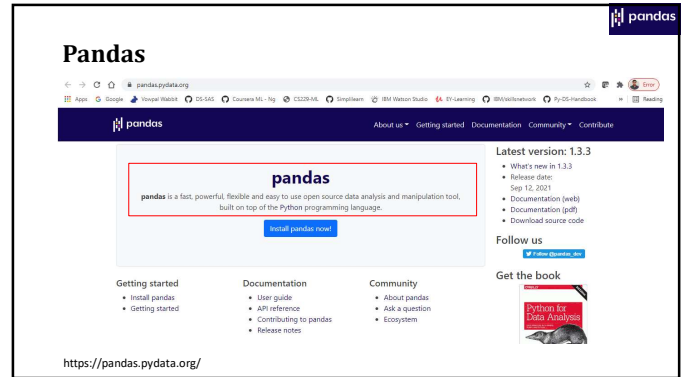
Search

<https://www.scipy.org/scipylib/index.html>

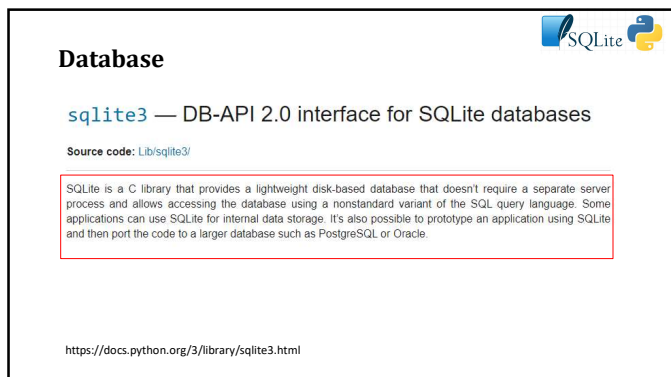
16



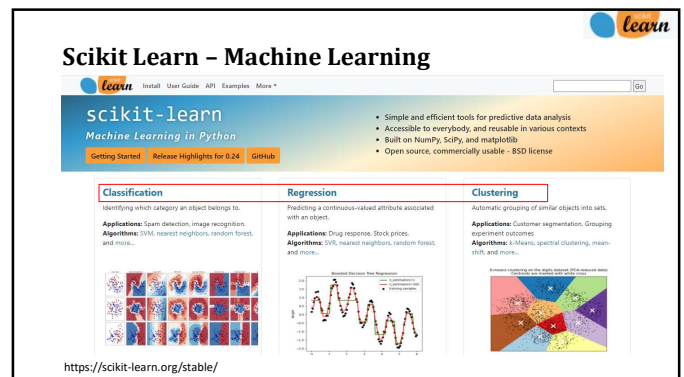
17



18



19



20



21

**Thank You**

Class material will be uploaded here:
https://github.com/soharabhossain/JoE_2021

22