Module 1.2.2

OpenCV Basics

Introduction to NumPy (Python)

Satya Mallick, Ph.D.
LearnOpenCV.com

Table of Contents

Introduction to the NumPy in Python-OpenCV

Python is a very popular general purpose programming language started by **Guido van Rossum**, primarily because of its simplicity and code readability, though it is slower than other languages like C/C++. Python enables the programmer to express their ideas in fewer lines of code and is an excellent language for rapid prototyping.

Another important feature of Python is that it can be easily extended with C/C++. This feature helps us write computationally intensive code in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as the original C/C++ code (since it is the actual C++ code working in the background) and second, it is very easy to code in Python. This is how OpenCV-Python works. It is a Python wrapper around the original C++ implementation.

The support of **Numpy** makes working with Python much easier. **Numpy** is a highly optimized Python library for numerical operations. All the OpenCV array structures are converted to-and-from Numpy arrays. So all operations that can be done in Numpy, you can combine it with OpenCV. Besides that, several other scientific libraries like SciPy, Matplotlib that support Numpy can be used.

OpenCV-Python is a great tool for fast prototyping of computer vision problems. A good knowledge on Numpy is required to write optimized code in OpenCV-Python.

NumPy adds support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

A **numpy array** is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The **shape** of an array is a tuple of integers giving the size of the array along each dimension.

This segment gives a brief introduction to the **NumPy** library.

```
# Import numpy
import numpy as np
# Create a 1 dimensional array
arr = np.array([5, 10, 15])
# To print the data type.
print arr.dtype
```

```
# Prints int64
# To print the shape of the array.
print arr.shape
# Prints "(3,)"
# Accessing the values stored and printing them.
print arr[0], arr[1], arr[2]
# Prints "5 10 15"
# Values of the array can be directly changed
arr[2] = 35
# To print the whole array.
print arr
# Prints "[5, 10, 35]"
# Create a 2 dimensional array
twoArr = np.array([[5,10],[15,20],[25,30]])
# To print the shape of the array.
print twoArr.shape
# Prints "(3,2)"
# Accessing the values stored and printing them.
print twoArr[0, 0], twoArr[0, 1], twoArr[1, 0]
# Prints "5 10 15"
# To print the whole array.
print twoArr
# Prints "[[ 5 10], [15 20],  [25 30]]"
```

Let us review some of the methods when NumPy is used in image operations:

Read an image

In OpenCV-Python environment, we use the function **cv2.imread()** to read an image. The image should be in the working directory or a full path of the image should be given.The first argument is the file name, the second argument is a flag which specifies the way image should be read. The different flags are described below:

- `cv2.IMREAD_COLOR` : Loads a color image. Any transparency of image will be neglected. It is the default flag.
- `cv2.IMREAD_GRAYSCALE` : Loads image in grayscale mode
- `cv2.IMREAD_UNCHANGED` : Loads image as such including alpha channel

Instead of these three flags, you can simply pass integers 1, 0 or -1 respectively.

In the Python code below, we saved the image file “bat.jpg” in our working directory. Then we intend to load the image in grayscale, thus we pass 0 as the second argument.

```python
1 # Import numpy and OpenCV
2 import numpy as np
3 import cv2
4 # Load a color image in grayscale
5 img = cv2.imread('bat.jpg',0)
```

Warning: Even if the image path is wrong, it won’t throw any error, but print img will give you None.

Assignment operation

Sometimes we may need to create a copy of an image, the simple assignment operation is illustrated in line 7 of the code below.

```python
1 # Import numpy and OpenCV
2 import numpy as np
3 import cv2
4 # Load a color image in grayscale
5 img = cv2.imread('bat.jpg',0)
6 # Assigning the image stored in numpy array img to another numpy array img_copy
7 img_copy = img
```

Copy part of an image

Sometimes we need to copy a part of the image and not an entire image. For example, in a face detection application, you may want to crop out the rectangular face region for further processing. The example below shows how to crop out a rectangular region.

```python
1 # Import numpy and OpenCV
2 import numpy as np
3 import cv2
4 # Load a color image in grayscale
5 img = cv2.imread('bat.jpg',0)
```

```python
# Copies from rows 100 to 150 and columns 100 to 250
cropped = img[100:150,100:250]
```

Find number of rows, columns, and channels

The **shape** method of a numpy array object can be used to find the number of rows (height), number of columns (width) and the number of channels of the image.

```python
# Import numpy and OpenCV
import numpy as np
import cv2
# Load a color image in BGR
img = cv2.imread('bat.jpg',1)
# Find the number of rows in the image
imgRows = img.shape[0]
# Find the number of columns in the image
imgCols = img.shape[1]
# Find the number of channels in the image
Channels = img.shape[2]
```

Warning: For a grayscale image, img.shape will return two parameters only- number of rows, number of columns.

Ones, zeros and identity

Like the C++ Mat class, NumPy also has a number of handy functions which helps create some basic matrices easily.

```python
# Import numpy
import numpy as np
# Create an array of length 5 with all elements as 0
a = np.zeros(5)
# Print out the array. Prints "[ 0.  0.  0.  0.  0.]" (Observe the decimal points)
print a
# Create an array of length 10 of all ones
```

```
a = np.ones(10)
# Print out the array.Prints "[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]"
print a
```

```
# Note for the functions above, specifying two dimensions (m,n) creates a 2D numpy array of m rows
and n columns
```

```
Create a 3x3 identity matrix
```

```
a = np.eye(3)
# Print out the matrix.Prints "[[ 1.  0.  0.], [ 0.  1.  0.] ,[ 0.  0.  1.]]" (in separate lines)
print a
```

```
# For the above functions the return type is float64 (can be found out by using a.dtype)
```

```
# Forces the datatype to be int64 in place of float64
```

```
a = np.ones(10, dtype=np.int64)
# Print out the array.Prints "[1 1 1 1 1 1 1 1 1 1]"
print a
```

Datatype Conversion

It is very common to convert the image format from unsigned int to floating point so that precision is not lost during calculations. The code snippet below illustrates data type conversion.

```
# Import numpy and OpenCV
```

```
import numpy as np
import cv2
```

```
# Load an color image in BGR
```

```
img = cv2.imread('bat.jpg',1)
# Prints uint8
print img.dtype
```

```
# Convert from uint8 to float32
```

```
img = np.float32(img)
# Prints float32
print img.dtype
```

```
# Convert from float32 to uint8
```

```
img = np.uint8(img)
# Prints uint8
print img.dtype
```

References and Further reading

http://cs231n.github.io/python-numpy-tutorial/

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

https://engineering.ucsb.edu/~shell/che210d/numpy.pdf