Overview of Python and the OpenCV Library for Image Processing

Nischal Regmi
Everest Engineering College

1 Introduction

Open Computer Vision (OpenCV) is an open source programming library that contains numerous functions for image processing and pattern recognition. OpenCV was initially developed by the Intel Corporation in 1999. OpenCV is written in C/C++, and provides interface to all major programming languages like C/C++, Java and Python. It is widely used in the industry as well in the academia. As a student, you can use OpenCV to get quick look at the outputs of several image processing algorithms you study. You may also use OpenCV if you are doing image processing research projects so that the coding of basic algorithms could be avoided.

So far, we have done IPPR lab using Java. Nonetheless, it would be better if you get acquainted with Python and the tools that professionals use for image processing. Python makes things simple (of course, with the price of slower execution.) I just want you to have an overview of OpenCV, which would be much easier using Python.

I won't be teaching the Python programming language in this tutorial. Fortunately, Python has a simple and elegant syntax that makes it very easy to learn. This is one of the reasons for its popularity. In addition, there is a very large collection of Python tutorial available online. I will also provide example codes that could be useful. However, we do not require much of programming. A knowledge of using Python libraries will be sufficient.

The official website for the Python language is `python.org`. To have a look at the Python Language Reference, visit: `https://docs.python.org/3/reference/`

In Python, we use several libraries like NumPy and Pandas. Reference manuals for such libraries are maintained by their respective creators.

2 Basic Examples Illustrating Python Syntax

Execute following examples to get an idea about Python language basics.

Example 1 Illustrating basic syntax of Python

```
# A very basic Python script
# Comment begins with hash

number1  = 23 #Unlike C++/Java, no line end-marker here
number2 = 17

#let's read another number from user
number3 = int(input('Enter an integer '))
# In the above line, we asked the user to enter an integer.
# By default whatever the user enters is string,
# so, we explicityly converted it to integer.
```

```
# But we would rarely ask the user to inupt anything in
# data mining modules.

# Now lets find the largest number.
# Python does not use brackets for control structures
# like if-else and function.
# Indentation (tab) plays the role of brackets.
maximum = number1
if number2 > maximum: #note the colon at the end
maximum = number2
if number3 > maximum:
maximum = number3

#now print the maximum
print('Maximum number is ', maximum)
```

Example 2 Lists in Python.

Lists in Python are flexible as they can store any data type and manipulated dynamic. Similar to C/C++, list index starts from 0.

```
#define a list and sort it
num1 = [17, 23, 10, 20] #initialized list
num1.sort()
print('Sorted List: ')
print(num1)

#define empty list, read 4 numbers, then sort
num2 = []
for i in range(0,4):
num2.append(int(input('Enter number: ')))
num2.sort()
print('Sorted List: ')
print(num2)
```

Example 3 This example introduces arrays and NumPy.

Arrays and related functions are implemented in Python's standard library NumPy (Numerical Python). This example illustrates how to import the numpy library and use it. Syntax and usage of any Python library is similar to this example.

NumPy arrays can be manipulated much faster using the functions provided in this library. However, for convenience, we will use loops to manipulate arrays. Similar to C/C++ and Python lists, NumPy arrays also begin with index value 0.

```
#import numpy library into the variabe np
import numpy as np

#declare numpy array of 5 elements
x = np.array([1,23,12,-4,8])
#another numpy array of 5 elements
y = np.array([34,21,200,83,0])
```

```python
# add two arrays element by element using
# a for-loop
s = np.empty(shape=(5)) #an array of size 5
for i in range(0,4):
s[i] = x[i]+y[i]
print(s)

#But as far as possible, don't use loops with
# numpy arrays. There are several built-in functions
# that execute faster
p = np.empty(shape=(5)) # an array of length 5
p = np.add(x,y)
print(p)

#we can declare a 2D arrays as this
A = np.array([[34,43, 45], [9, 3, 1], [12, 47, 8]]) #3 by 3 array
B = np.array([[-1,-4, 56], [213, 2, 6], [19, 10, 1]])
C = np.empty(shape=(3,3)) #an array of 3 rows and 3 cols
C = np.add(A,B)
print(C)
```

3 OpenCV Examples

Example 1 Convolution using OpenCV spatial filter function.

```python
import cv2 as cv
import numpy as np

# Read the input image as grayscale
# second argument 0 --> read as grayscale
# second argument 1 --> read as color (this is default)

img = cv.imread('Input image path',0)
cv.imshow('Original Image',img)
cv.waitKey()

#create a box kernel with 1's --> for averaging
kernel = np.ones((5,5),np.float32)/25

#apply the 2D convolution operator
result = cv.filter2D(img,-1,kernel)
cv.imshow('After convolution',result)
cv.waitKey()

#save the result -- (if you want)
cv.imwrite('output image path',result)
```

Note that in the above program, we have used a 5×5 matrix of 1/25 as the kernel. In the function 'filter2D' function, the second argument is -1, which is to assure that the data type of the output matrix is same as the input matrix. In this example, the input matrix is of type 'uint8', i.e. eight-bit unsigned integer. Since we are averaging the pixels, we are sure that the result will not negative, so, there would not be any problem if we agree with 'uint8' for the output matrix. However, if we could have negative values after the convolution, we need to change the output matrix's type to, for example, 32-bit floating point type. This could be achieved by

```
result = cv.filter2D(img,cv.CV_32F,kernel)
```

You need to refer the details from official OpenCV documentation about the possible data types for matrices.

Example 2 Sobel gradient using OpenCV's convolution function.

```
import cv2 as cv
import numpy as np

img = cv.imread('Input image path',0)

# Sobel gradient for x direction
# Since we are using convolution, we need to rotate the
# matrix

kx = np.array([[1,2,1],[0,0,0],[-1,-2,-1]])
gx = cv.filter2D(img,cv.CV_32F,kx)


#Sobel gradient for y direction (rotated)
ky = np.array([[1,0,-1],[2,0,-2],[1,0,-1]])
gy = cv.filter2D(img,cv.CV_32F,ky)

#calculate the magnitude
M = np.add(np.abs(gx),np.abs(gy))


#scale the magnitude to [0,255] range
mn = np.min(M)
mx = np.max(M)
M = 255*(M-mn)/(mx-mn)

#display the gradient image
cv.imshow('Image gradient',M.astype(np.uint8))
cv.waitKey()

#save the result -- (if you want)
#cv.imwrite('output image path',M)
```

Example 3 Binary thresholding using OpenCV

4 Some Problems You Would Like to Ponder Upon

1. Display the edges in an image using the Laplacian operator. For this, you need to first find the Laplacian by convolving the Laplacian kernel with the image. Then you need to threshold the result.

2. In lab sheet 3, you had an image 'Messi_N.jpg' that has been corrupted by sinusoidal interference. Use the Discrete Fourier transform and other appropriate functions from the OpenCV library to remove the sinusoidal interference based on notch filters.

3. Find edges using Sobel's gradient, and make it clearer by rescaling and thresholding. As you have already noticed in the earlier labs, the gradient image usually contains hundreds unwanted of unwanted dots. Use morphological erosion and dilation operators to clean the dots.

4. Explore segmentation algorithms in the OpenCV library and implement programs that

   (a) Recognizes and extracts human faces from an image.

   (b) Recognizes and extracts vehicle license plates (number plates) from an image.

   (c) Recognizes and extracts individual characters from the vehicle number plates extracted in above problem.

5. **Mini-Project**: How accurate are the segmentation algorithms of the OpenCV library as per your observations in problem 4? For this, you can focus on any of the three sub-problems, say, (c). Then you have to gather images with vehicle license plates with different orientations, brightness, noises, etc. and analyze the performance.