Unit-V
Image Processing Using OpenCV Python

P. Balamurugan, Assistant Professor
PG & Research Department of Computer Science
Government Arts College, Coimbatore -641018
Email: spbalamurugan@rediffmail.com

Outline of Lecture

- What is IPython?
- Python Environment
- Python Basic operations
- Image manipulation using IPython

What is IPython?

- **Python** is a general-purpose interpreted, interactive, functional, structured, object-oriented, and high-level programming language.
- It was created by Guido van Rossum during 1985-90.
- Used as a scripting language, compiled to byte-code for building large applications.

What is IPython?(Contd…)

- It supports automatic garbage collection.
- Easily integrated with C, C++, COM, ActiveX, CORBA, and Java.
- It provides features like tab completion and easier access to help.

Applications of IPython

- **Easy-to-learn**
- **Easy-to-read**
- **Easy-to-maintain**
- **A broad standard library**
- **Interactive Mode**
- **Portable**
- **Extendable**
- **Databases**
- **GUI Programming**
- **Scalable**

Flavors of Python

- ***Cpython -*** The base of all these implementation is Cpython or more formally known as python. It is written in C and is a bytecode interpreter.
- ***Jpython -*** Jython, earlier known as Jpython (or known to be successor of Jpython) is an implementation of the Python programming language which is designed to run on the Java Platform. It has compiler which compiles your python code into Java bytecode
- ***IronPython -*** open-source implementation of the Python programming language which is tightly integrated with the .NET Framework.
- ***PyPy -*** It is a Python interpreter and just-in-time compiler. It is written in Python itself. Actually PyPy is written in a language called Rpython

Flavors of Python

- ***Rpython*** – Ruby Python : A bridge between the Ruby and Python interpreters. It embeds a running Python interpreter in the Ruby applications.
- ***Pythonxy*** - Pronounced as Python x-y and mostly written as Python(X,Y). It is nothing but a scientific Python distribution. Python added with scientific and engineering related packages is your python(x,y).
- ***Anaconda Python*** - it is distribution for large-scale data processing, predictive analytics, and scientific computing. This implementation basically focuses large scale of data.
- ***Stackless Python*** - Not depending on the C call stack for its own stack

Ipython Environment

Python version – 3.8.5, Ipython version – 7.12

Spyder(Python 3.8) IDE consists following parts :

- Menu Bar
- Tool Bar
- Console Window
    1. Execute Commands
    2. List Command History
- Workspace Window
    1. List files Tab
    2. Variable Explorer Tab
    3. Plots Tab
    4. Help Tab

Ipython Basic Operations

- Command Prompt : In[1]
- Ctrl + D : Quit Console
- Ctrl + C : Cancel command
- Arrow keys : Show Ipython session history
- ? Help
- ?? Detail help
- Tab completion access : Auto completion of command
- Example:

```
In[1] : 1+1
Out[1] : 2

In[2] : print(1+1)
2
```

Image Manipulation Using Ipython

```
In[2] :  import matplotlib.pyplot as plt
In[3] : img= plt.imread(‘  ‘)
In[4] : plt.imshow(img)
In[5] : plt.show()
In[6] : img.shape
In[7] : plt.imshow(img,cmap='gray')
```

Accessing Parts of an Array

- Access and change
- Rows and Columns of an Array
- Elements of an array using Slicing and Striding
- Example:

```
A = ([1, 2, 3, 4, 5])
C = ([[1, 2, 3, 4, 5],
      [6, 7, 8, 9, 10],
      [11, 12, 13, 14, 15],
      [16, 17, 18, 19, 20],
      [21, 22, 23, 24, 25]])
```

Accessing Parts of an Array

- Python programming supports negative indexing of arrays I. This means the index value of I -1 gives the last element I -2 gives the second to last element of an array.
- Slicing of an array is done to access parts of an array I.
  - Slicing syntax is [start:stop]
- Striding uses the step value to jump between the elements in an array I.
  - Striding syntax is [start:stop:step]

Conditional Statements

- if block : **if condition** is used to decide whether to execute the statements in the **if block** or not.
- Syntax:

```
if <condition>:
#statements
```

- if/else block : When the test condition is true, it will execute the body of **if**. If the condition is **False**, body of the **else** is executed. Note that, **if** and **else** statements end with a colon. This denotes the beginning of the code block of **True** or **False** condition.
- Syntax:

```
if <condition>:
#True block statements
else:
#False block statements
```

Conditional Statements(Contd…)

- Example:

```
num = int(input())
if num % 2 == 0:
        print ('Even')
else:
        print ('Odd')
```

Conditional Statements(Contd…)

- if/elif/else block: **elif** are same as **if/else statements.** in **if/else,** only one condition is checked. But in **elif statement** multiple conditions can be checked.
- Syntax:

```
if <condition-1>:
#statements when cond-1 True
elif <condition-2>:
#statements when cond-2 True
else:
#statements when all conditions Fail
```

Conditional Statements(Contd…)

- Example:

```
if a > 0:
print ("positive")
elif a < 0:
print ("negative")
else:
print ("zero")
```

Conditional Statements

- Ternary conditional statement: Ternary operator allows to test a condition in a single line replacing the multiline **if-else**.

  It can reduce the code size and increase the readability of the code.

- Syntax :

  [A] if <condition> else [B]

- Example :

  max= a if a > b else b

- Pass statement

Conditional Statements(Contd…)

- *Pass statement* : It acts as a *null* operation (i.e) nothing happens when it executes. It works as a placeholder for block of code. It is used in a code block where actual code implementation is not known yet.
- Example :

```
if a%2 == 0:
        print('even')
else:
        pass
```

Loop Statements

- for loop
- while loop
- break
- continue
- pass

Loop Statements

- While loop : execute a set of statements as long as a condition is true. When the condition becomes false, program control passes to the line immediately after the loop.
- Syntax:

```
while <condition>:
#True block statements
#update condition variable
#statements after while loop
```

- Example

```
i = 1
while i<10:
print (i*i)
i += 2
```

Loop Statements(Contd…)

- for loop : iterates over a list or any other sequential data type.
- Syntax:

```
for <id in range(start,end,step)>:
#block statements
#statements after for loop
```

- Example

```
for n in range(1, 10, 2):
print (n*n)
```

Loop Statements(Contd…)

- pass : It is just a placeholder. It is used for the sake of completion of blocks, that do not have any code within them.
- Example :

```
for n in range(2, 10, 2):
pass
```

- break : It is used to break the innermost loop.
- Example :

```
for letter in 'python':
if letter == 'h':
break
print ('Current Letter :', letter)
```

Loop Statements(Contd…)

- continue : Rejects all the remaining statements in the current iteration of the loop..
- Example :

```
for n in range(1, 10, 2):
if n % 3 == 0:
continue
print (n*n)
```

References

- https://www.slideshare.net/CristinaPrezBenito/simultaneous-smoothing-and-sharpening-of-color-images.
- https://www.javatpoint.com/digital-image-processing-tutorial.
- https://www.tutorialspoint.com/dip/index.html.

TEXT BOOKS

1) Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing", Second Edition, PHI/Pearson Education.
2) Alexander M., Abid K., "OpenCV-Python Tutorials", 2017.

REFERENCE BOOKS

1) B. Chanda, D. Dutta Majumder, "Digital Image Processing and Analysis", PHI, 2003.
2) Nick Efford, "Digital Image Processing a practical introducing using Java", Pearson Education, 2004.