Python for Image Processing

Pedro Garcia

http://www.sawp.com.br

Department of Computer Science – University of Brasilia – Brazil

2011, September, 8
Tópicos
Python as Language

- High-level abstractions language
  semi-functional + Object Oriented
Python as Language

- High-level abstractions language
  semi-functional + Object Oriented
- Interpreted with a JIT compiler (relatively fast).
Python as Language

- High-level abstractions language
  - semi-functional + Object Oriented
- Interpreted with a JIT compiler (relatively fast).
- Designed for simplicity:
  - easy to read + easy to write = maintainable
Python as Language

- High-level abstractions language
  - semi-functional + Object Oriented
- Interpreted with a JIT compiler (relatively fast).
- Designed for simplicity:
  - easy to read + easy to write = maintainable
- Scalable
Python as Language

- High-level abstractions language
  semi-functional + Object Oriented
- Interpreted with a JIT compiler (relatively fast).
- Designed for simplicity:
  easy to read + easy to write = maintainable
- Scalable
- Extensible (C or Fortran for speed)
## Python vs Matlab

<table>
<thead>
<tr>
<th>Feature</th>
<th>Python</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenSource</td>
<td>✓</td>
<td>Ø</td>
</tr>
<tr>
<td>Free</td>
<td>✓</td>
<td>Ø</td>
</tr>
<tr>
<td>Multiplatform</td>
<td>✓</td>
<td>Ø</td>
</tr>
<tr>
<td>Fast Memory Management</td>
<td>✓</td>
<td>Ø</td>
</tr>
<tr>
<td>GUI compatibility</td>
<td>✓</td>
<td>Ø</td>
</tr>
<tr>
<td>Easily distributable</td>
<td>✓</td>
<td>Ø</td>
</tr>
<tr>
<td>Interchangeable with non-numerical applications</td>
<td>✓</td>
<td>Ø</td>
</tr>
<tr>
<td>Standalone scripting</td>
<td>✓</td>
<td>Ø</td>
</tr>
<tr>
<td>Native matrix operators and functions</td>
<td>Ø</td>
<td>✓</td>
</tr>
</tbody>
</table>
Tópicos
Unlike the M language, Python hasn’t native support for matrix and others numerical operations.
Unlike the M language, Python hasn’t native support for matrix and others numerical operations.

However, is possible get all the MATLAB features combining a libraries set:
Resources for numerical computing in Python

- Unlike the M language, Python hasn’t native support for matrix and others numerical operations.
- However, is possible get all the MATLAB features combining a libraries set:
  - SciPy
Resources for numerical computing in Python

- Unlike the M language, Python hasn’t native support for matrix and others numerical operations.
- However, is possible get all the MATLAB features combining a libraries set:
  - SciPy
  - Matplotlib/PyLab
Unlike the M language, Python hasn’t native support for matrix and others numerical operations. However, is possible get all the MATLAB features combining a libraries set:

- SciPy
- Matplotlib/PyLab

To add image processing resources in Python is necessary to install the Python Imaging Library (required by the above libraries).
SciPy

- OpenSource library of numerical tools.
SciPy

- OpenSource library of numerical tools.
- MATLAB-like syntax.
SciPy

- OpenSource library of numerical tools.
- MATLAB-like syntax.
- The main module, NumPy, provide a lot of functions and operators for multidimensional arrays, providing a clean syntax and rely on LAPACK for efficient linear algebra computations.
SciPy

- OpenSource library of numerical tools.
- MATLAB-like syntax.
- The main module, NumPy, provide a lot of functions and operators for multidimensional arrays, providing a clean syntax and rely on LAPACK for efficient linear algebra computations.
- BSD license.
SciPy

- Modules
SciPy

### Modules

- **constants**: physical constants and conversion factors
- **cluster**: hierarchical clustering, vector quantization, K-means
- **fftpack**: Discrete Fourier Transform algorithms
- **integrate**: numerical integration routines
- **interpolate**: interpolation tools
- **io**: data input and output
- **lib**: Python wrappers to external libraries
- **linalg**: linear algebra routines
- **misc**: miscellaneous utilities (e.g. image reading/writing)
- **optimize**: optimization algorithms including linear programming
- **signal**: signal processing tools
- **sparse**: sparse matrix and related algorithms
- **spatial**: KD-trees, nearest neighbors, distance functions
- **special**: special functions
- **stats**: statistical and regression functions
- **weave**: tool for writing C/C++ code as Python strings
Matplotlib/PyLab

- OpenSource plotting library.
Matplotlib/PyLab

- OpenSource plotting library.
- Compatible with SciPy, since its a NumPy extension.
Matplotlib

- OpenSource plotting library.
- Compatible with SciPy, since it's a NumPy extension.
- The OO library design allows embedding the plots using generic GUI, independent of a specific library (Qt, WinAPI, GTK).
Matplotlib/PyLab

- OpenSource plotting library.
- Compatible with SciPy, since its a NumPy extension.
- The OO library design allows embedding the plots using generic GUI, independent of a specific library (Qt, WinAPI, GTK).
- Compatible with libcairo, generating plots and image as vector-graphics-based (SVG). Matlab doesnt supports this natively.
Matplotlib/PyLab

- OpenSource plotting library.
- Compatible with SciPy, since it's a NumPy extension.
- The OO library design allows embedding the plots using generic GUI, independent of a specific library (Qt, WinAPI, GTK).
- Compatible with libcairo, generating plots and image as vector-graphics-based (SVG). Matlab doesn't support this natively.
- The **pylab** package combines the plotting library with NumPy into a single namespace, resembling the MATLAB experience.
Simple Example: SciPy + Matplotlib

```
#!/usr/bin/env python
#
# Make a legend for specific lines.
from pylab import *

# Make a legend for specific lines.
from pylab import *

t1 = arange(0.0, 2.0, 0.1)
t2 = arange(0.0, 2.0, 0.01)

l1, = plot(t2, exp(-t2))
l2, l3 = plot(t2, sin(2 * pi * t2), '--go', t1, log(1 + t1), '. '.
l4, = plot(t2, exp(-t2) * sin(2 * pi * t2), 'rs-'.

legend( (l2, l4), ('oscillatory', 'damped'), 'upper right', shadow=True)
xlabel('time')
ylabel('volts')
title('Damped oscillation')
show()
```
Simple Example: SciPy + Matplotlib

Damped oscillation

oscillatory
damped
Simple Example: SciPy + Matplotlib

```
#!/usr/bin/env python

import numpy as np
from matplotlib.pyplot import figure, show, rc

# radar green, solid grid lines
rc('grid', color='#316931', linewidth=1, linestyle='--')
rc('xtick', labelsize=15)
rc('ytick', labelsize=15)

# force square figure and square axes looks better for polar, IMO
fig = figure(figsize=(8, 8))
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8], polar=True, axisbg='#d5de9c')

r = np.arange(0, 3.0, 0.01)
theta = 2*np.pi*r
ax.plot(theta, r, color='#ee8d18', lw=3, label='a line')
ax.plot(0.5*theta, r, color='blue', ls='--', lw=3, label='another line')
ax.legend()

show()
```
Simple Example: SciPy + Matplotlib
Simple Example: SciPy + Matplotlib

```python
#!/usr/bin/env python
import numpy
import pylab

z = lambda x,y: (1 - x + x ** 3 + y ** 5) * numpy.exp(-(x ** 2 + y ** 2))
x = numpy.arange(-3.0, 3.0, 0.025)
y = numpy.arange(-3.0, 3.0, 0.050)
(X, Y) = pylab.meshgrid(x, y)
Z = z(X, Y)
image = pylab.imshow(Z, interpolation='bilinear', cmap=pylab.cm.Spectral)
v = numpy.arange(-1.2, 1.6, 0.2)
cset = pylab.contour(Z, v, linewidths=2, cmap=pylab.cm.hot)
pylab.clabel(cset, inline=True, fmt='%1.1f', fontsize=10)
pylab.colorbar(image)
pylab.axis('off')
pylab.title('$z=(1-x+x^3+y^5) e^{-(x^2+y^2)}$')
```
Simple Exemple: SciPy + Matplotlib

\[ z = (1-x + x^3 + y^5) e^{- (x^2 + y^2)} \]
Python Image Library (PIL)

- OpenSource plotting library.
Python Image Library (PIL)

- OpenSource plotting library.
- Support for opening, reading and writing image files in formats PNG, JPEG, GIFF, TIFF and BMP.
Python Image Library (PIL)

- OpenSource plotting library.
- Support for opening, reading and writing image files in formats PNG, JPEG, GIFF, TIFF and BMP.
- Used internally by the mentioned libraries.
Tópicos
SciPy supports natively all images operations provided by MATLAB.
SciPy supports natively all images operations provided by MATLAB.

```
help(scipy.ndimage)
```
to check functions for filtering, interpolation, morphology, segmentation, distance transform and more.

Others Libraries

- Mahotas:
  - advanced algorithms for watershed, convex hull computation, Haralick textures, local binary patterns, Zernike moment etc.
- PyMorph:
  - several functions for morphological, segmentation and pattern recognition operations.
SciPy supports natively all images operations provided by MATLAB.

`help(scipy.ndimage)` to check functions for filtering, interpolation, morphology, segmentation, distance transform and more.

However, more than basic functions provided by SciPy (and MATLAB) may be necessary.
Others Libraries

- SciPy supports natively all images operations provided by MATLAB.

- `help(scipy.ndimage)` to check functions for filtering, interpolation, morphology, segmentation, distance transform and more.

- However, more than basic functions provided by SciPy (and MATLAB) may be necessary.

- Others specialized and compatible with NumPy libraries are available in OpenSource:
SciPy supports natively all images operations provided by MATLAB.

help(scipy.ndimage) to check functions for filtering, interpolation, morphology, segmentation, distance transform and more.

However, more than basic functions provided by SciPy (and MATLAB) may be necessary.

Others specialized and compatible with NumPy libraries are available in OpenSource:

- **Mahotas:** advanced algorithms for watershed, convex hull computation, Haralick textures, local binary patterns, Zernike moment etc.
SciPy supports natively all images operations provided by MATLAB.

`help(scipy.ndimage)` to check functions for filtering, interpolation, morphology, segmentation, distance transform and more.

However, more than basic functions provided by SciPy (and MATLAB) may be necessary.

Others specialized and compatible with NumPy libraries are available in OpenSource:

- **Mahotas**: advanced algorithms for watershed, convex hull computation, Haralick textures, local binary patterns, Zernike moment etc.
- **PyMorph**: several functions for morphological, segmentation and pattern recognition operations.
In real problems, image processing techniques are associated with other techniques for solution (machine learning, artificial intelligence, Statistical techniques, etc).
Another argument

- In real problems, image processing techniques are associated with other techniques for solution (machine learning, artificial intelligence, Statistical techniques, etc).
- Considering this, the solution SciPy + Python is better because has more resources with other libraries.
In real problems, image processing techniques are associated with other techniques for solution (machine learning, artificial intelligence, Statistical techniques, etc).

Considering this, the solution SciPy + Python is better because has more resources with other libraries.

**PyMVPA**: Multivariate Pattern Analysis in Python.
Another argument

- In real problems, image processing techniques are associated with other techniques for solution (machine learning, artificial intelligence, Statistical techniques, etc).
- Considering this, the solution SciPy + Python is better because has more resources with other libraries.
- **PyMVPA**: Multivariate Pattern Analysis in Python.
- **Milk**: Machine Learning Toolkit for Python.
Another argument

- In real problems, image processing techniques are associated with other techniques for solution (machine learning, artificial intelligence, Statistical techniques, etc).
- Considering this, the solution SciPy + Python is better because it has more resources with other libraries.
- **PyMVPA**: Multivariate Pattern Analysis in Python.
- **Milk**: Machine Learning Toolkit for Python.
- **scikits.learn**: machine learning in Python.
Another argument

- In real problems, image processing techniques are associated with other techniques for solution (machine learning, artificial intelligence, Statistical techniques, etc).
- Considering this, the solution SciPy + Python is better because it has more resources with other libraries.
- **PyMVPA**: Multivariate Pattern Analysis in Python.
- **Milk**: Machine Learning Toolkit for Python.
- **scikit-learn**: Machine learning in Python.
- **PyCUDA**: SciPy compatible library to run on massively parallel GPU environments.
Tópicos
Examples: SciPy vs MATLAB

- Central forums, blogs, websites, codes.

MATLAB Central: http://www.mathworks.com/matlabcentral/index.html
SciPy Central: http://scipy-central.org/
Examples: SciPy vs MATLAB

- Central forums, blogs, websites, codes.
  - MATLAB Central:
Examples: SciPy vs MATLAB

Central forums, blogs, websites, codes.

- MATLAB Central:
- SciPy Central: http://scipy-central.org/
MATLAB: Resizing with Bilinear Interpolation

```matlab
function [rIm] = imresize_matlab(im, n)
    im = int32(im);
    [low_h, low_w, channels] = size(im);
    h = (low_h - 1) * n;
    l = (low_w - 1) * n;
    for i=1:low_h
        for j=1:low_w
            rIm(1 + (i - 1) * n,1 + (j - 1) * n,:) = im(i,j,:);
        end
    end
    for i=1:h
        for j=1:l
            rest1 = rem(i-1, n);
            rest2 = rem(j-1, n);
            d00=rIm(ceil(i/n)*n-n+1,ceil(j/n)*n-n+1,:);
            d10=rIm(ceil(i/n)*n-n+1+n,ceil(j/n)*n-n+1,:);
            d01=rIm(ceil(i/n)*n-n+1,ceil(j/n)*n-n+1+n,:);
            d11=rIm(ceil(i/n)*n-n+1+n,ceil(j/n)*n-n+1+n,:);
            x = rest1;
            y = rest2;
            dx=x/n;
            dy=y/n;
            b1=d00;
            b2=d10-d00;
            b3=d01-d00;
            b4=d00-d10-d01+d11;
            rIm(i,j,:)=b1+b2*dx+b3*dy+b4*dx*dy;
        end
    end
    rIm = uint8(rIm);
    return
```
SciPy: Resizing with Bilinear Interpolation

```python
def imresize_python(arr, newsize, interp='bilinear', mode=None):
    arr = asarray(arr)
    im = toimage(arr, mode=mode)
    if type(newsize) is types.IntType:
        newsize = float(newsize)
    if type(newsize) is types.FloatType:
        newsize = (im.size[0] * newsize, im.size[1] * newsize)
    else:
        newsize = (newsize[1], newsize[0])
    func = {'nearest': 0, 'bilinear': 2, 'bicubic': 3, 'cubic': 3}
    im = im.resize(newsize, resample=func[interp])
    return fromimage(im)
```
Examples: Converting to Safe Colors

MATLAB

```matlab
function Om = safe_color(Im)
    Im = uint16(Im);
    Im = ceil((6.0 / 255.0) * Im);
    R = uint8(Im(:,:,1));
    G = uint8(Im(:,:,2));
    B = uint8(Im(:,:,3));
    R(R == 1) = hex2dec('00');
    R(R == 2) = hex2dec('33');
    R(R == 3) = hex2dec('66');
    R(R == 4) = hex2dec('99');
    R(R == 5) = hex2dec('CC');
    R(R == 6) = hex2dec('FF');
    G(G == 1) = hex2dec('00');
    G(G == 2) = hex2dec('33');
    G(G == 3) = hex2dec('66');
    G(G == 4) = hex2dec('99');
    G(G == 5) = hex2dec('CC');
    G(G == 6) = hex2dec('FF');
    B(B == 1) = hex2dec('00');
    B(B == 2) = hex2dec('33');
    B(B == 3) = hex2dec('66');
    B(B == 4) = hex2dec('99');
    B(B == 5) = hex2dec('CC');
    B(B == 6) = hex2dec('FF');
    Om(:,:,1) = R;
    Om(:,:,2) = G;
    Om(:,:,3) = B;
    Om = uint8(Om);
end
```

SciPy

```python
def safecolors(Im):
    Im = ceil((6.0 / 255.0) * Im)
    R = Im[:,:,0]
    G = Im[:,:,1]
    B = Im[:,:,2]
    R = (R == 1).choose(R, 0x00)
    R = (R == 2).choose(R, 0x33)
    R = (R == 3).choose(R, 0x66)
    R = (R == 4).choose(R, 0x99)
    R = (R == 5).choose(R, 0xCC)
    R = (R == 6).choose(R, 0xFF)
    G = (G == 1).choose(G, 0x00)
    G = (G == 2).choose(G, 0x33)
    G = (G == 3).choose(G, 0x66)
    G = (G == 4).choose(G, 0x99)
    G = (G == 5).choose(G, 0xCC)
    G = (G == 6).choose(G, 0xFF)
    B = (B == 1).choose(B, 0x00)
    B = (B == 2).choose(B, 0x33)
    B = (B == 3).choose(B, 0x66)
    B = (B == 4).choose(B, 0x99)
    B = (B == 5).choose(B, 0xCC)
    B = (B == 6).choose(B, 0xFF)
    Om = zeros((Im.shape[0], Im.shape[1], 3))
    Om[:,:,0] = R
    Om[:,:,1] = G
    Om[:,:,2] = B
    return Om
```
Examples: Converting to Safe Colors
Examples: Color threshold

MATLAB

```matlab
1 function Om = color_threshold(grayscale, rr, color)
2    outR = grayscale;
3    outG = grayscale;
4    outB = grayscale;
5    R = color(1);
6    G = color(2);
7    B = color(3);
8    for i=rr(1):rr(2);
9        outR(outR == i) = R;
10       outG(outG == i) = G;
11       outB(outB == i) = B;
12    end
13    Om(:,:,1) = outR;
14    Om(:,:,2) = outG;
15    Om(:,:,3) = outB;
16 end
```

SciPy

```python
1 def color_threshold(im, rr, cc):
2    (outR, outG, outB) = (im, im, im)
3    (R, G, B) = (cc[0], cc[1], cc[2])
4    Om = zeros((im.shape[0], im.shape[1], 3))
5    for i in xrange(rr[0], rr[1]):
6        outR = (outR == i).choose(outR, R)
7        outG = (outG == i).choose(outG, G)
8        outB = (outB == i).choose(outB, B)
9    (Om[:,:,0], Om[:,:,1], Om[:,:,2]) = (outR, outG, outB)
10   return Om
```
Examples: Color Threshold
OpenCV

- Originally an Intel research initiative.
OpenCV

- Originally an Intel research initiative.
- Cross-platform.
OpenCV

- Originally an Intel research initiative.
- Cross-platform.
- OpenSource.
OpenCV

- Originally an Intel research initiative.
- Cross-platform.
- OpenSource.
- Mostly employed for **real time image processing**.
OpenCV

- Originally an Intel research initiative.
- Cross-platform.
- OpenSource.
- Mostly employed for **real time image processing**.
- Focus on Computer Vision (CV).
OpenCV

- Originally an Intel research initiative.
- Cross-platform.
- OpenSource.
- Mostly employed for **real time image processing**.
- Focus on Computer Vision (CV).
- Written in C, ensuring fast and portable code.
OpenCV

- Originally an Intel research initiative.
- Cross-platform.
- OpenSource.
- Mostly employed for **real time image processing**.
- Focus on Computer Vision (CV).
- Written in C, ensuring fast and portable code.
- Multiple languages bindings, such as OpenCV.Net, EmguCV, C++, **PyOpenCV**, JOpenCV (Java and Scala), GPUCV and more.
Image processing has been by development in C/C++ and MATLAB.
Image processing has been by development in C/C++ and MATLAB.

MATLAB is used as a high level platform for prototyping and testing.
Image processing has been by development in C/C++ and MATLAB.

- MATLAB is used as a high level platform for prototyping and testing.
- C/C++ is designed for optimized applications that require performance.
Image processing has been by development in C/C++ and MATLAB.

- MATLAB is used as a high level platform for prototyping and testing.
- C/C++ is designed for optimized applications that require performance.

- The Python design enable wrappers with both advantages.
Image processing has been by development in C/C++ and MATLAB.

- MATLAB is used as a high level platform for prototyping and testing.
- C/C++ is designed for optimized applications that require performance.

The Python design enable wrappers with both advantages.

- The official Python wrappers are compatible with NumPy.
Image processing has been by development in C/C++ and MATLAB.

- MATLAB is used as a high level platform for prototyping and testing.
- C/C++ is designed for optimized applications that require performance.

The Python design enable wrappers with both advantages.

- The official Python wrappers are compatible with NumPy.
- It allow combine all vantages of OpenCV with SciPY (multiprocessing, GPU processing for NumPy, etc).
Python for Image Processing

Python and OpenCV

Image processing has been by development in C/C++ and MATLAB.
- MATLAB is used as a high level platform for prototyping and testing.
- C/C++ is designed for optimized applications that require performance.

The Python design enable wrappers with both advantages.
- The official Python wrappers are compatible with NumPy.
- It allow combine all vantages of OpenCV with SciPy (multiprocessing, GPU processing for NumPy, etc).
- As Python is extensible in C/C++/Fortran, the bottlenecks functions can be written in these languages.
Examples: Color threshold

cvMat to NumPy

```python
1 import numpy
2
3 def cvMat2ndarray(opencv_matrix):
4     return numpy.asarray(opencv_matrix)
```

NumPy to cvMat

```python
1 import cv
2
3 def ndarray2cvMat(numpy_array):
4     return cv.fromarray(numpy_array)
```
Tópicos
References and Resources

- http://scipy.org/
- http://packages.python.org/mahotas/
- http://www.tramy.us/
- PEP8 - Style Guide for Python Code
- http://opencv.willowgarage.com/
- http://www.scipy.org/PerformancePython
- http://numpy.scipy.org/
- http://ipython.scipy.org/moin