Painless Application Development with 3D Slicer and Python

Demian Wassermann
LMI 09/2008
Main Idea

Convince you that the development model in Slicer 3 can be made easy by using Python. Moreover that you’ll be able to directly prototype your software in Slicer 3.
Main Idea

Convince you that the development model in Slicer 3 can be made easy by using Python. Moreover that you’ll be able to directly prototype your software in Slicer 3.

I’m presenting the work performed principally by Daniel Blezek and Luca Antiga (with small collaborations of D. Allen, S. Pieper and me).
Using C++ tools

- Optimized for computers or (some) humans
- Low level (unless you’re using VTK/ITK)
- S loooow development cycle
- Non interactive, whereas scientific work is inherently exploratory
Using C++ tools

However.......

• Millions of Lines of Code have been produced and tested
• Excellent performance
• In fact we need to work with these, not replace them
Higher Level Tools

- Mathematica & Maple, something on the side
- IDL & Matlab: extremely popular
  - Great interactivity, visualization and extensions
  - Languages not suited for medium-scale projects
  - Used for prototyping which leads to a lot of code rewriting
Mixed approaches

- Develop bits and pieces of your code in C++, Matlab
- Build enormous scripts chaining tools, saving and loading files to communicate
- Use other tools to visualize your results

Huge context switching overhead
Where does Python Stand

- Free (BSD license) and portable
- Interactive
- Clear syntax and suitable for large-scale projects
- Object Oriented model, but not mandatory
- Very comprehensive library
- Lots of numerical analysis oriented packages
- Simple C++, C, FORTRAN integration
From Matlab to Python

<table>
<thead>
<tr>
<th>Matlab</th>
<th>Python / Numpy</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(2:5)</td>
<td>a[1:4]</td>
</tr>
<tr>
<td>a(1:end)</td>
<td>a(0:)</td>
</tr>
<tr>
<td>a'</td>
<td>a.T</td>
</tr>
<tr>
<td>a(a&gt;.5)</td>
<td>a[a&gt;.5]</td>
</tr>
<tr>
<td>[V,D]=eig(a)</td>
<td>V,D=linalg.eig(a)</td>
</tr>
</tbody>
</table>

and there are lot of packages for optimization, image processing, statistics, learning, etc.

http://www.scipy.org/NumPy_for_Matlab_Users
Actual Development Cycle

• Prototype your stuff in Matlab (with some divine help to handle medical images)

• Check that it works (or something like that)

• Reprogram everything in ITK/VTK/3D Slicer (with some divine help from Luis, Steve and other martyrs)

• Spend a lot of time compiling - debugging - compiling - debugging waiting for 3D Slicer to run......

• Deploy and deliver.
3D Slicer/Python
Development

• Build a module skeleton in Python
• Write an XML description of your algorithm’s interface
• Write a single execute procedure
• Run Slicer 3D
• Code, press “apply”, test, correct
  • Eventually locate performance problems and recode specific parts in C++
• Deploy and Deliver
Interactive demo 1

>>> from Slicer import slicer

- Import required libraries
Interactive demo 1

```python
>>> from Slicer import slicer

>>> node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1')
```

- Import required libraries
- Get the node from the Scene
Interactive demo 1

```python
>>> from Slicer import slicer

>>> node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1')
>>> nodeID = node.GetImageData()
>>> nodeIDArray = nodeImageData.ToArray()
```

- Import required libraries
- Get the node from the Scene
- Get the array representing the node
Interactive demo 1

```python
>>> from Slicer import slicer

>>> node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1')
>>> nodeID = node.GetImageData()
>>> nodeIDArray = nodeImageData.ToArray()

>>> nodeIDArray[nodeIDArray>150] = 150
```

- Import required libraries
- Get the node from the Scene
- Get the array representing the node
- Clip image intensities
Interactive demo 1

```python
>>> from Slicer import slicer

>>> node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1')
>>> nodeID = node.GetImageData()
>>> nodeIDArray = nodeImageData.ToArray()

>>> nodeIDArray[nodeIDArray>150] = 150
>>> node.Modified()
```

- Import required libraries
- Get the node from the Scene
- Get the array representing the node
- Clip image intensities
- Slicer refresh
Interactive demo 2
Interactive demo 2

```python
>>> from scipy import ndimage
>>> from Slicer import slicer
```

- Import required libraries
Interactive demo 2

```python
>>> from scipy import ndimage

>>> from Slicer import slicer

>>> node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1')
```

- Import required libraries
- Get the node from the Scene
Interactive demo 2

```python
>>> from scipy import ndimage
>>> from Slicer import slicer

>>> node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1')
>>> nodeID = node.GetImageData()
>>> nodeIDArray = nodeImageData.ToArray()
```

- Import required libraries
- Get the node from the Scene
- Get the array representing the node
Interactive demo 2

```python
from scipy import ndimage
from Slicer import slicer

node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1')
nodeID = node.GetImageData()
nodeIDArray = nodeImageData.ToArray()

nodeIDArray[:] = ndimage.median_filter(nodeIDArray, 2)
```

- Import required libraries
- Get the node from the Scene
- Get the array representing the node
- median filtering and assignment
Interactive demo 2

```python
>>> from scipy import ndimage
>>> from Slicer import slicer

>>> node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1'
>>> nodeID = node.GetImageData()
>>> nodeIDArray = nodeImageData.ToArray()

>>> nodeIDArray[:] = ndimage.median_filter(nodeIDArray, 2)

>>> node.Modified()
```

- Import required libraries
- Get the node from the Scene
- Get the array representing the node
- median filtering and assignment
- Slicer refresh
Interactive demo 2

>>> from scipy import ndimage
>>> from Slicer import slicer

>>> node = slicer.MRMLScene.GetNodeByID('vtkMRMLScalarVolumeNode1')
>>> nodeID = node.GetImageData()
>>> nodeIDArray = nodeImageData.ToArray()

>>> nodeIDArray[:,] = ndimage.median_filter(nodeIDArray, 2)

>>> node.Modified()

Override array contents

- Import required libraries
- Get the node from the Scene
- Get the array representing the node
- median filtering and assignment
- Slicer refresh
Scripted Module 1
Scripted Module 1

How to start a module
Scripted Module 1

How to start a module

- Module description

XML Module signature

```xml
<executable>
  <category>Demo Scripted Modules</category>
  <title>Masked median filtering</title>
  <description>
  Perform median filtering over a masked section of an image
  </description>
  <version>1.0</version>
  <documentation-url></documentation-url>
  <license></license>
  <contributor>Demian Wassermann</contributor>

Python function signature

```python
def Execute(
```
Scripted Module 1

How to start a module

- Module description
- Parameters
  - Input image

XML Module signature

```xml
<parameters>
  <label>IO</label>
  <description>Input/output parameters</description>

  <image type = "scalar">
    <name>inputVolume</name>
    <longflag>inputVolume</longflag>
    <label>Input Image</label>
    <channel>input</channel>
    <description>Input image to be filtered</description>
  </image>
</parameters>
```

Python function signature

```python
def Execute(
    inputVolume = "",
```
Scripted Module 1

How to start a module

- Module description
- Parameters
  - Input image
  - Median filter radius

XML Module signature

```xml
<integer>
  <name>medianFilterRadius</name>
  <longflag>medianFilterRadius</longflag>
  <label>Radius of the median filter</label>
  <default>2</default>
  <step>1</step>
  <channel>input</channel>
  <constraints>
    <minimum>2</minimum>
    <maximum>100</maximum>
  </constraints>
</integer>
```

Python function signature

```python
def Execute(
    inputVolume = "",
    medianFilterRadius = 0,
```
Scripted Module 1

How to start a module

- Module description
- Parameters
  - Input image
  - Median filter radius
  - Label Image

XML Module signature

```xml
<image type="label">
  <name>inputMaskVolume</name>
  <longflag>inputMaskVolume</longflag>
  <label>Input Mask Volume</label>
  <channel>input</channel>
  <description>Input mask to work on it</description>
</image>
```

Python function signature

```python
def Execute(
    inputVolume = "",
    medianFilterRadius = 0,
    inputMaskVolume = "",
```
Scripted Module 1

How to start a module

- Module description
- Parameters
  - Input image
  - Median filter radius
  - Label Image
  - Label to use

XML Module signature

```xml
<integer>
  
  <name>labelToUse</name>
  
  <longflag>labelToUse</longflag>
  
  <label>Label to use for the mask</label>
  
  <default>1</default>
  
  <step>1</step>
  
  <channel>input</channel>
  
  <constraints>
    
    <minimum>0</minimum>
    
    <maximum>255</maximum>
    
  </constraints>

</integer>
```

Python function signature

```python
def Execute(
    inputVolume = "",
    medianFilterRadius = 0,
    inputMaskVolume = "",
    labelToUse = 1,
)```
Scripted Module 1

How to start a module

- Module description
- Parameters
  - Input image
  - Median filter radius
  - Label Image
  - Label to use
  - Output Image

XML Module signature

```xml
<image type = "scalar">
  <name>outputFilteredVolume</name>
  <longflag>outputFilteredVolume</longflag>
  <label>Output Image</label>
  <channel>output</channel>
  <description>Image that was median filtered</description>
</image>
```

Python function signature

```python
def Execute(
    inputVolume = "",
    medianFilterRadius = 0,
    inputMaskVolume = "",
    labelToUse = 1,
    outputFilteredVolume = ""
):
```
Things available in Python/Numpy/Scipy

- Available at www.scipy.org
- Open Source BSD Style License
- Over 30 svn “committers” to the project

CURRENT PACKAGES

- Special Functions (scipy.special)
- Signal Processing (scipy.signal)
- Image Processing (scipy.ndimage)
- Fourier Transforms (scipy.fftpack)
- Optimization (scipy.optimize)
- Numerical Integration (scipy.integrate)
- Linear Algebra (scipy.linalg)
- Input/Output (scipy.io)
- Statistics (scipy.stats)
- Fast Execution (scipy.weave)
- Clustering Algorithms (scipy.cluster)
- Sparse Matrices (scipy.sparse)
- Interpolation (scipy.interpolate)
- More (e.g. scipy.odr, scipy.maxentropy)

[Oliphant]
Scripted Module 2

On Real Life
Questions?
Questions?

Thanks