Painless Application Development with 3D Slicer and Python h

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)
- Sloooow 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

Matlab	Python / Numpy
a(2:5)	a[1:4]
a(1:end)	a(0:)
a’	a.T
a(a>.5)	a[a>.5]
[V,D]=eig(a)	V,D=linalg.eig(a)

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

Interactive demo 1

```
>>> from Slicer import slicer
```

- Import required libraries

Interactive demo 1

```
>>> from Slicer import slicer

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

- Import required libraries
- Get the node from the Scene

Interactive demo 1

```
>>> 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

```
>>> 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

```
>>> 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

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

- Import required libraries

Interactive demo 2

```
>>> 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

```
>>> 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

```
>>> 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

```
>>> 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

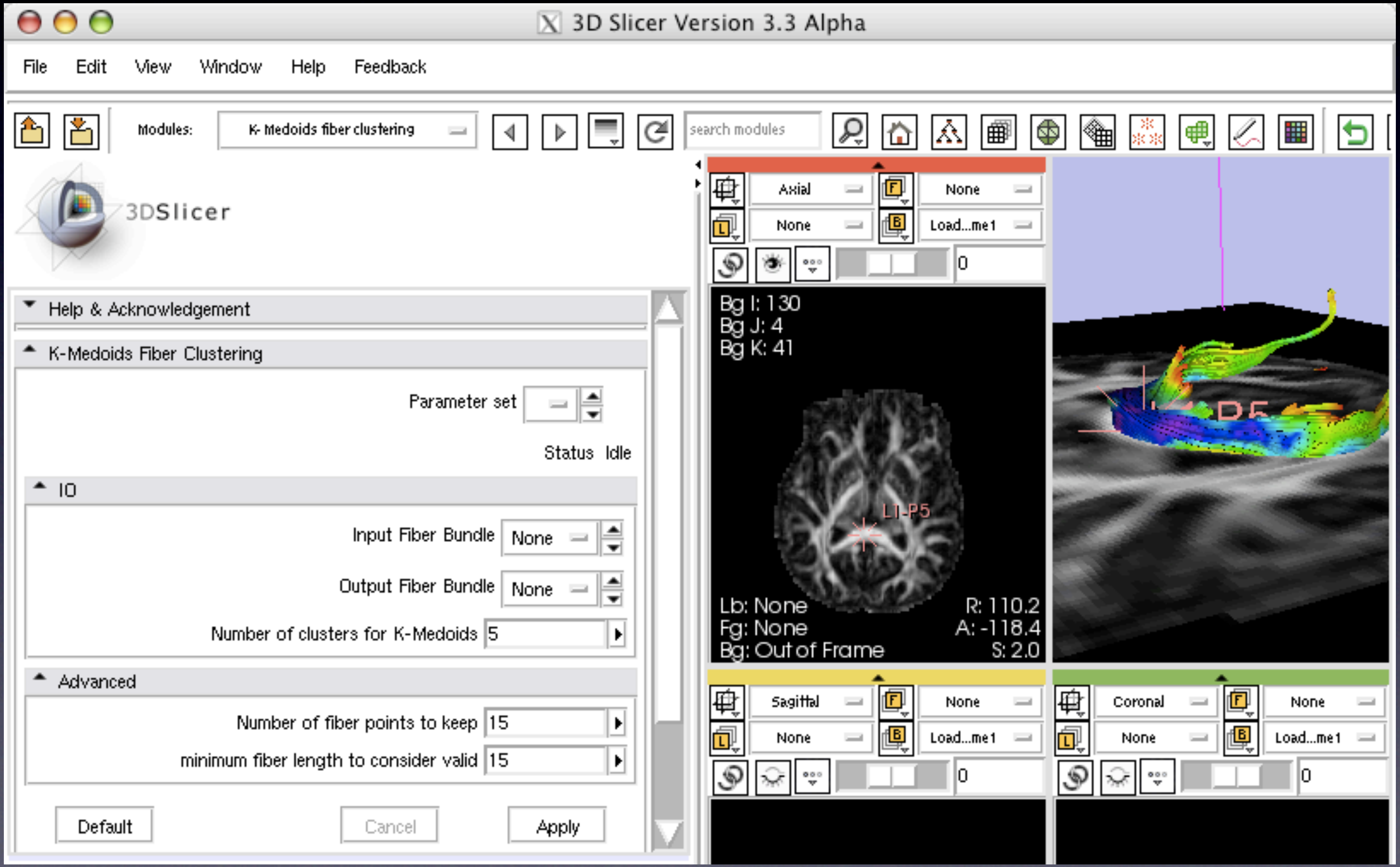
Interactive demo 2

```
>>> 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()
```

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

Override array contents

Scripted Module 1



Modules: K-Medoids fiber clustering

K-Medoids Fiber Clustering

Parameter set

Status Idle

IO

Input Fiber Bundle None

Output Fiber Bundle None

Number of clusters for K-Medoids 5

Advanced

Number of fiber points to keep 15

minimum fiber length to consider valid 15

Bg I: 130
Bg J: 4
Bg K: 41

Lb: None
Fg: None
Bg: Out of Frame

R: 110.2
A: -118.4
S: 2.0

Load...me1

Scripted Module 1

How to start a module

Scripted Module 1

How to start a module

- Module description

XML Module signature

```
<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

```
def Execute(\
```

Scripted Module 1

How to start a module

- Module description
- Parameters
  - Input image

XML Module signature

```
 <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>
```

Python function signature

```
def Execute(\
     inputVolume = "",\
```

Scripted Module 1

How to start a module

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

XML Module signature

```
<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

```
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

```
    <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

```
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

```
<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

```
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

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

  </parameters>

</executable>
```

Python function signature

```
def Execute(\
     inputVolume = "",\
     medianFilterRadius = 0,\
     inputMaskVolume = "",\
     labelToUse = 1,\
     outputFilteredVolume = ""\
     ):
```

Scripted Module 1

On Real Life

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