Running Matlab® functions in 3D Slicer using MatlabBridge

Andras Lasso[1], Kevin Alexander[2], Christopher Jechel[2], Kevin Wang[3], John Schreiner[4], Gabor Fichtinger[1]

[1]Laboratory for Percutaneous Surgery, School of Computing, Queen's University; [2]Department of Physics, Queen's University; [3]Radiation Medicine Program, Princess Margaret Cancer Centre, University Health Network; [4]Cancer Centre of S.E. Ontario

Introduction

- Matlab® is a powerful and widely used software environment for the development of computational algorithms.
- However, **there are challenges using Matlab® based software in the clinical environment**: Matlab® medical image data import and export, processing and visualization modules are limited, and in-house developed code is difficult to maintain locally or exchange between institutions.
- 3D Slicer (www.slicer.org) is a free, open source software platform that has comprehensive medical image analysis and visualization capabilities and it can be further enhanced and customized by extensions developed in the Python and/or C++ programming languages.
- **Goal: Implement a software bridge that allows running Matlab® functions directly from within 3D Slicer.**
  - Provide rich medical image visualization and pre/post-processing capabilities during Matlab® algorithm development and optimization
  - Make the developed Matlab® algorithms readily usable for clinicians in the 3D Slicer application.

Methods

System overview

- Main features of the MatlabBridge (Fig. 1)
  - Starts the Matlab® process automatically when a function execution is requested
  - Sends input data, runs the algorithm, and receives processing results
  - Uses OpenIGTLink protocol for transferring input/output parameters
  - Uses files to exchange bulk data (image volumes, surface models, etc.)

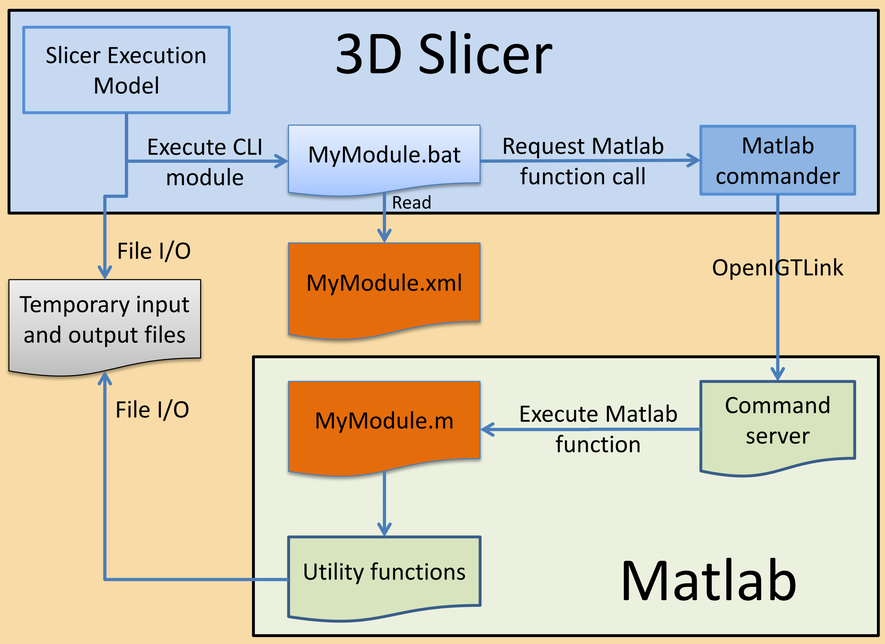

Figure 1. Architecture overview of the MatlabBridge extension; 3D Slicer and Matlab® communicate through file input/output and OpenIGTLink commands.

- Each user-developed Matlab® function has a corresponding MatlabBridge module in Slicer, with an automatically-generated graphical user interface.
- The graphical user interface of the MatlabBridge module (input/output images, points, scalar value selectors, sliders, editboxes, checkboxes, etc.) are defined in a short XML file.
- As a use-case and way of testing, a so called gamma comparison method was implemented, which compares planned versus measured radiotherapy dose distributions. The gamma function (Low and Dempsey, Med. Phys., 2003) is a widely used dosimetry comparison tool, which incorporates both dose difference and distance-to-agreement information into a single metric.

Results

- The MatlabBridge module is available for the latest version of 3D Slicer, it can be installed and configured in a few minutes using the extension manager in 3D Slicer.
- Available for free, on Windows, Linux, and Mac platforms
- The extension includes a helper tool (MatlabModuleGenerator module) for generating skeleton modules that the developer can customize and extend.

Example: Evaluation for 3D gamma dose computation

- User interface of a 3D gamma Matlab® function and the result of a dose comparison are shown in Fig. 2.
- Note that no modifications of the original Matlab® functions were needed to run them in 3D Slicer.
- Dose volume inputs for the algorithms were imported by the SlicerRT extension (www.SlicerRT.org). Standard 3D Slicer volume reslicing and volume rendering were used for visualization and analysis of the results.

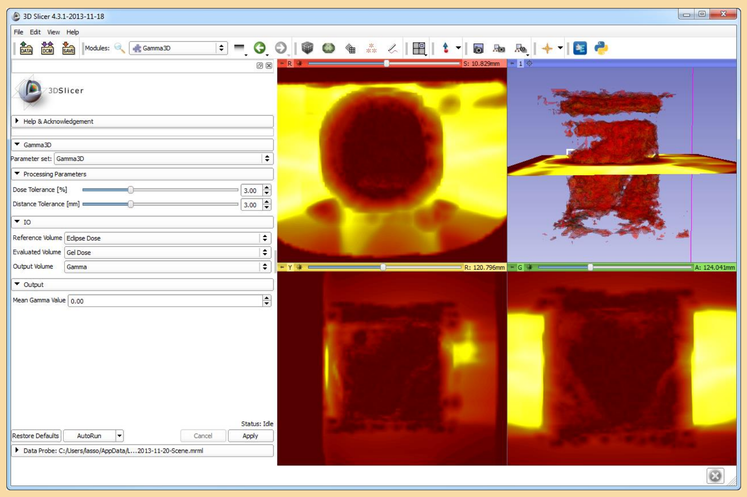

Figure 2. Comparison result of dose volumes computed by the treatment planning system (Eclipse™) for a prostate IMRT treatment and a gel dosimeter measurement of the dose delivery read by optical CT.

```
function outputParams=Gamma3D(inputParams)
% 3D gamma dose comparison

ref=cli_imageread(inputParams.refvolume);
eval=cli_imageread(inputParams.evalvolume);
Reference = double(ref.pixelData);
Evaluated = double(eval.pixelData);
DoseC = inputParams.dosetol/100;
DistCr = inputParams.disttol;

[GAMMA] = Gamma3dAniso(Evaluated,Reference,DoseC,DistCr);

outputParams.MEAN=mean(GAMMA(:));

gammaVOL = ref;
gammaVOL.pixelData=GAMMA;
cli_imagewrite(inputParams.outputvolume, gammaVOL);
```

Figure 3. Matlab® function executed by 3D Slicer.

```
<executable>
  <category>Matlab</category>
  <title>Gamma3D</title>
  <description>Compute gamma comparison in
Matlab</description>

<parameters>
  <label>Processing Parameters</label>
  <description>Parameters for the processing</description>
  <double>
    <label>Dose Tolerance [%]</label>
    <description>Voxels below this value …</description>
    <longflag>dosetol</longflag>
    <default>3</default>
    <constraints>
      <minimum>0</minimum> <maximum>10</maximum>
      <step>0.5</step>
    </constraints>
  </double>
  <double>
    <label>Distance Tolerance [mm]</label>
    <description>Voxels below this value …</description>
    <longflag>disttol</longflag>
    <default>3</default>
    <constraints>
      <minimum>0</minimum> <maximum>10</maximum>
      <step>0.5</step>
    </constraints>
  </double>
</parameters>

<parameters>
  <label>IO</label>
  <description>Input/output parameters</description>
  <image>
    <label>Reference Volume</label>
    <description>Input volume to be filtered</description>
    <longflag>refvolume</longflag>
    <channel>input</channel>
  </image>
  <image>
    <label>Evaluated Volume</label>
    <description>Input volume to be filtered</description>
    <longflag>evalvolume</longflag>
    <channel>input</channel>
  </image>
  <image>
    <label>Output Volume</label>
    <description>Output filtered</description>
    <longflag>outputvolume</longflag>
    <channel>output</channel>
  </image>
</parameters>

<parameters>
  <label>Output</label>
  <description>Matlab command outputs</description>
  <double>
    <label>Mean Gamma Value</label>
    <name>MEAN</name>
    <channel>output</channel>
    <default></default>
  </double>
</parameters>
```

Figure 4. Matlab® module descriptor: describes the inputs and outputs of the function, used for automatic graphical user interface generation.

Conclusion

Using the MatlabBridge extension, functions implemented in Matlab® can be run from within 3D Slicer through a convenient graphical user interface. Practical application has been demonstrated in radiation therapy use case.

Detailed documentation, tutorial, examples are available at:

http://www.slicer.org/slicerWiki/index.php/Documentation/4.3/Extensions/MatlabBridge

Acknowledgement

This work was supported through the Applied Cancer Research Unit program of Cancer Care Ontario with funds provided by the Ontario Ministry of Health and Long-Term Care, and CIHR. Gabor Fichtinger was funded as a Cancer Ontario Research Chair.