IRONPYTHON SCRIPTING IN A RADIATION THERAPY TREATMENT PLANNING SYSTEM


Rickard Holmberg, RaySearch Laboratories
TALK OVERVIEW

- Radiation Therapy – a primer
  - The optimization problem
- The Treatment Planning System (TPS)
  - Components
- Our earlier scripting approach
- The scripting in the current product
  - Implementation
  - Use cases
  - How to get users started
- Future directions
ABOUT RAYSEARCH AND ME

• RaySearch Laboratories
  • started in 2001 from the Karolinska Institute,
  • now has ~60 developers working in Stockholm, Sweden.
• I moved from the Karolinska Institute to RaySearch in 2007.
  • Background in applied physics, linux systems programming.
  • Using numpy and friends since about 2004.
• RaySearch originally made libraries for optimization of radiation therapy plans, used in e.g. Philips’ and Siemens’ radiotherapy treatment planning systems.
• In 2010 we released a complete treatment planning system for radiotherapy.
RADIATION THERAPY – A PRIMER
Radiation therapy aims to cure cancer by killing tumours with ionizing radiation.

Bremsstrahlung photon radiation @ 6 - 25 MV is most common, but higher energies and other particles are also used, both electrons and protons are common.
• Beam laterally conformed using collimators.
• Tumours are often located deep in the body. The radiation must somehow be delivered mostly to the tumour, avoiding the healthy tissue around it.
• Multiple beams focused at target.
• Uniformity in target volume by modulating each beam.
RADIATION TREATMENT WORKFLOW
BASIC CASE

1. Patient is imaged in a CT scanner
2. Organs and tumours are delineated
3. Treatment goals are specified
4. Treatment is optimized
5. Optimized treatment is verified (QA)
6. Patient is treated

1. This is done externally to the Treatment Planning System.
2. On the images, either via manual or semiautomatic means.
3. Done jointly by an oncologist and a physicist to ensure tumor control.
4. For delivery on a specific treatment machine unit.
5. By comparing actual measurements on a phantom to expected results.
6. In several treatment sessions, normally about 30.
OPTIMIZATION OF THE THERAPY PROBLEM

- Search space: Either
  - Fluence “pixel” values (convex), or
  - machine parameters (tungsten leaf positions, amount of radiation from each beam, etc.) (not convex, only locally)

- Objective functions:
  - Mean dose to volume (convex)
  - Percentile rank of a specific volume to receive a given dose, e.g. ≤10% of the eye should receive ≥10 Gy (not convex)

- Constraints:
  - physical machine limits and
  - clinical constraints, such as maximum dose to organs.
OUR TREATMENT PLANNING SYSTEM
• Treatment planning system for external-beam radiotherapy
• GUI, persistency, data model with rules, all in C#
• Solver, dose calculation, other algorithms in C++ (and OpenCL)
• Features:
  • Image registration, rigid and deformable
  • Automatic markup of organs
  • Treatment optimization and dose calculation
  • Multi-criteria optimization
IMAGE REGISTRATION – RIGID AND DEFORMABLE
MODEL-BASED SEGMENTATION
MULTI-CRITERIA OPTIMIZATION - NAVIGATION
SCRIPTING IN PREVIOUS PRODUCTS
ACCESSING .NET APPLICATIONS USING IRONPYTHON

- IronPython is a Python implementation alongside CPython, PyPy and Jython.
- Targets the CLR (Microsoft .NET / Mono).
- Regular CLR binaries have enough information to script against.

```python
import clr
clr.AddReference("RaySearch.CorePlatform.Framework")

from RaySearch.CorePlatform.Framework import SessionState
from RaySearch.CorePlatform.DomainModels.PatientDB import OrganType

patientDB = SessionState.Instance.CreateDomainState("PatientDB")

for patient in patientDB.Root.Patients:
    plan = patient.Studies[0].MainTreatmentCase.Plans[0]
    dgrs = plan.Doses[0].DoseGridRois
    print patient.Name.DisplayName, 
          [dgr.RoiVolumeDistribution.TotalVolume
           for dgr in dgrs
           if dgr.OrganType == OrganType.Kidney]
```
WHY CAN’T WE DO IT LIKE THAT, THEN?

- Use in clinical environment, for real patients
- Safety and regulatory reasons make restrictions necessary:
  - Disqualifies unrestricted access to data model (e.g. plan approval)
- Same process
  - memory corruption
  - no numpy for 64-bit IronPython
  - Tricky exception handling, separating application errors from scripting errors
- …and the list goes on…
SCRIPTING IN RAYSTATION
SCRIPTING IMPLEMENTATION

- Very narrow network protocol:
  - Get globals.
  - Get/set members.
  - Invoke methods and receive return values.
- Proxy objects in the interpreter
- The network protocol can be configured to present a SOAP interface, but is currently configured for a binary WCF-only interface.
SCRIPTING USE-CASES

Clinical work

• Automating tedious (and error-prone) manual operations:
  • Clinic-specific protocols.
  • Clinic-specific evaluation tools.

Research

• Data-mining stored plans.
• Re-planning studies.
• Alternative objective functions, optimizers, volume definitions, etc.
BEGINNING SCRIPTING IN RAYSTATION – RECORDING SCRIPTS

```python
plan = get_current("Plan")
beam_set = get_current("BeamSet")

with CompositeAction('Add Optimization Function'):
    retval_0 = plan.PlanOptimizations[0].AddOptimizationFunction(
        FunctionType="MaxEud", RoiName="Brain")

retval_0.DoseFunctionParameters.DoseLevel = 5000

plan.PlanOptimizations[0].ResetOptimization()
plan.PlanOptimizations[0].RunOptimization(ComputeBeamDoses=True)
beam_set.ComputeDose(ComputeBeamDoses=True, ComputePointDoses=False)
```
BEGINNING SCRIPTING IN RAYSTATION – BROWSING THE DATA MODEL

- Data model state can be examined using a live State Viewer.
- Includes documentation on methods available to scripts.
IronPython – lags CPython in library support.

Though a preliminary NumPy and SciPy version for IronPython exists, that branched off in 2010.

According to Charles Harris, “I think it is pining for the fjords.”

Not all methods available in the GUI are scriptable (yet).

Quite network intensive – one request per member access.

Currently only the plain ipy interpreter – would be nice with either a spyder-like or ipython notebook-like UI.

IronLab (www.ironlab.net) has a pretty neat UI for IronPython, with prepackaged NumPy.NET and SciPy.NET, plus a custom plotting library.