Pyplanscoring Documentation

Release 0.1.0

Victor Gabriel Leandro Alves, D.Sc.

May 16, 2018

Table of Contents

CHAPTER 1

Introduction

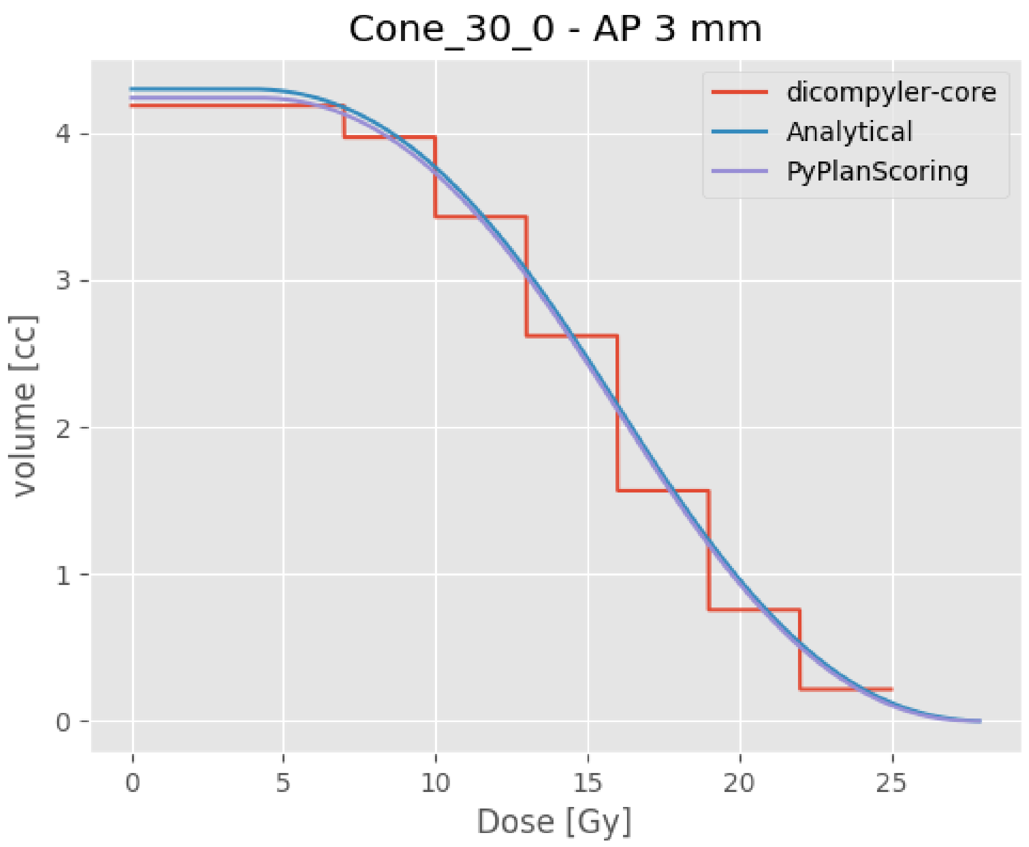
PyPlanScoring (PPS) package is software that calculates Dose Volume Histogram (DVH) for plans generated by most commercially available Treatment Planning Systems (TPSs). PPS was built extending the dicompyler-core package: to perform structure volume oversampling using dose trilinear interpolation, mainly in small structures. PPS was built initially to evaluate plans submitted to the international radiotherapy plan competitions organized by Radiation Knowledge.

CHAPTER 2

Motivation

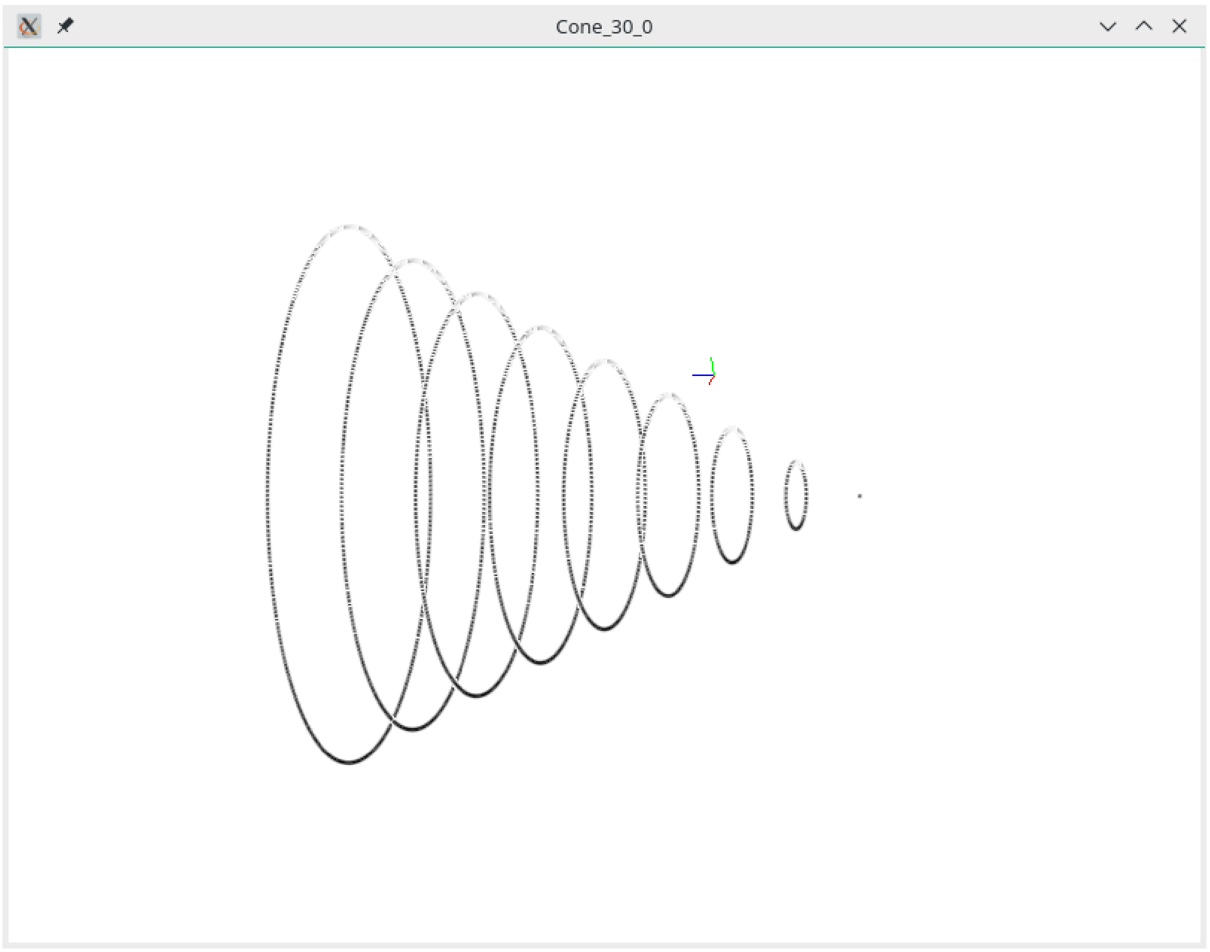
To develop a Python API to estimate DVHs and evaluate constraints metrics. The main contribution is to implement structure volume up-sampling/oversampling. Then, It improves the calculation accuracy on DVHs on small structures. This API makes use of high-performance numerical libraries, such as Scipy, Numpy and Numba which supports compilation of Python to run on either CPU or GPU hardware. Most of critical code was implemented from scratch, e.g, dose-contour rasterization, DVH metrics following the recommendations of AAPM Report No. 263 - Standarizing Nomenclatures in Radiation Oncology (2018).

The picture bellow shows a DVH calculated using a small cone, 3 mm slice size on 1D linear dose gradient.



The dicompyler-core algorithm does not provide an accurate representation of this DVH. On the other hand, PPS algorithm can be set to calculate oversampling using voxels down to 0.1 mm ^{3} or less. Its results are in close agreement to analytical values.

3D representation of an analytical cone structure with 3 mm slice thickness:



The collection of DVH test data is gently provided by Canis Lupus LLC on the link. More details here.

CHAPTER 3

Where to from here?

If you're new to this library, you can visit the *user manual* page. If you're already familiar with this library, or you want to dive straight in, you can jump to the Sphinx AutoAPI Index. You can also see the contents in the sidebar.

3.1 User Manual

The main class in this package is the `PyPlanScoringAPI`. Learning a few things about this class can be very useful in using this library. This section attempts to document some common things about this object.

3.1.1 Installing

You can get `pyplanscoring` from its current source on GitHub, to get all the latest and greatest features. `pyplanscoring` is under active development, and many new features are being added. However, note that the API is currently unstable at this time.

```
git clone https://github.com/victorgabr/pyplanscoring.git
cd ./pyplanscoring
python setup.py install
```

3.1.2 Requirements

PyPlanscoring as was designed to be a modern python library. It targets long-term support using the newest features in data science. It is recommended to set up a python 3.6 or higher environment because typing annotations will be adopted gradually.

Installing python using Anaconda/Miniconda environment and conda-forge channel is highly recommended.

```
python>=3.6.0
pillow>=5.1.0
pydicom>=1.0.2
numba>=0.37.0
numpy>=1.12.1
scipy>=1.0.0
pandas>=0.22.0
quantities>=0.12.1
```

Optional

These packages have to be installed to activate visualization and multiprocessing capabilities.

```
matplotlib>=2.0.0
joblib>=0.11
```

```
vispy>=0.5.3
```

conda environment

It is possible to install all dependencies using:

```
conda install -c conda-forge pillow pydicom numba numpy scipy pandas quantities
matplotlib joblib vispy
```

After installing Anaconda/Miniconda Python 3.6.

GPU computing

There is an experimental support to GPU computing `pyplanscoring.core.gpu_code`. DVH calculation kernels were written in Numba. It supports CUDA GPU programming by directly compiling a restricted subset of Python code into CUDA kernels and device functions following the CUDA execution model.

```
conda install cudatoolkit
```

3.1.3 Getting Started

This section is going to provide basic applications of `pyplanscoring` API. The main functionality is encapsulated on `PyPlanScoringAPI` class.

Its constructor receives paths to DICOM-RTStructure and RT-DOSE files. These files are commonly exported by most of radiotherapy planning Systems - TPS.

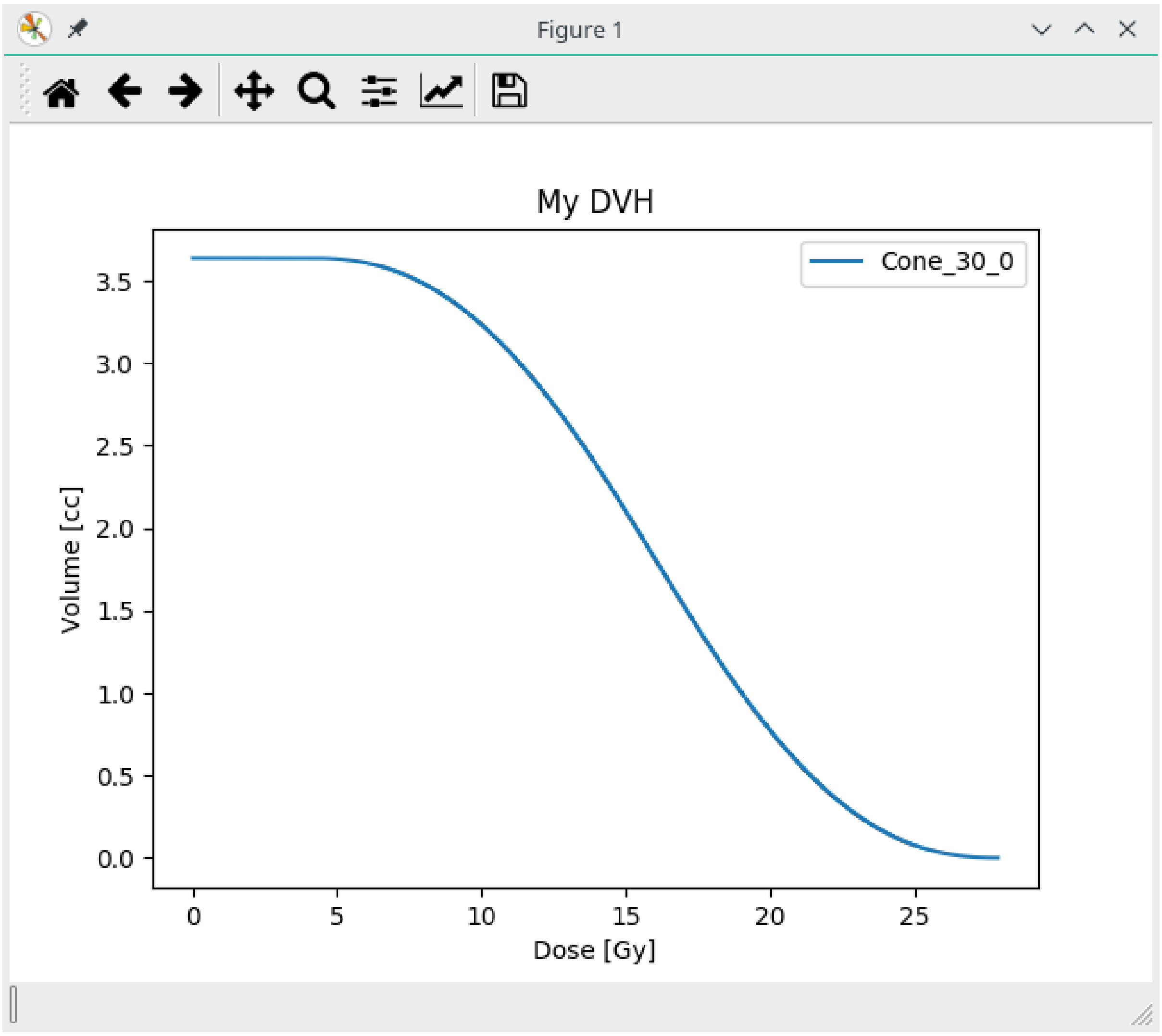
The method `PyPlanScoringAPI.get_structure_dvh` returns a dictionary with a calculated cummulative DVH in absolute volumes [cc].

It is also possible to save calculated DVHs in JavaScript Object Notation (JSON) files. Just using the class `IOHandler`.

```
from pyplanscoring import PyPlanScoringAPI, plot_dvh, plot_dvhs, IOHandler
# DVH calculation use-case
# RS file
rs_file = 'RT-Structure.dcm'
# RD file
rd_file = 'RT-DOSE.dcm'

pp = PyPlanScoringAPI(rs_file, rd_file)

#calculation parameters
end_cap_size = 1.5  # mm
calc_grid = (0.1, 0.1, 0.1)   # mm3

# calculating one structure DVH using roi_number
dvh = pp.get_structure_dvh(roi_number=2, end_cap=end_cap_size,
calc_grid=calc_grid)

# plotting DVH
plot_dvh(dvh, 'My DVH')

# calculating DVH from all strucures in RT-structure file - no oversampling
dvhs = pp.calc_dvhs(verbose=True)

# Plotting all DVHs in relative volumes
plot_dvhs(dvhs, 'PyPlanScoring')

# saving results in JSON text
obj = IOHandler(dvhs)
output_file_path = 'plan_dvhs.dvh'
obj.to_json_file(output_file_path)
```

This example shows a result using oversampling by setting the calculation grid size (dx, dy, dz) in mm.

Excample result:



That's it! You can move on to the *user manual* to see what part of this library interests you.

3.1.4 DVH metrics

This section provides extended aplications of `pyplanscoring` API. PyPlanscoring has methods to extract dose-volume histogram metrics with regular expression operators It improves the ability to use computer algorithms to automate calculation of DVH metrics. This functionality is provided by the package `pyplanscoring.constraints`.

It is possible to handle a single structure DVH using the class `DVHMetrics`

The recommended nomenclature is described on section 9 of the AAPM Report No. 263.

The user must set metrics in string format. Example:

```
from pyplanscoring import PyPlanScoringAPI, DVHMetrics

# DVH calculation use-case
# RS file
rs_file = 'RT-Structure.dcm'
# RD file
rd_file = 'RT-DOSE.dcm'
```

```

pp = PyPlanScoringAPI(rs_file, rd_file)

#calculation parameters
calc_grid = (0.2, 0.2, 0.2)  # mm3

# calculating one structure DVH using roi_number
dvh = pp.get_structure_dvh(roi_number=2, calc_grid=calc_grid)

# getting DVH metrics
dvh_metrics = DVHMetrics(dvh)
metrics = ['Min[Gy]',
           'Mean[Gy]',
           'Max[Gy]',
           'D99%[Gy]',
           'D95%[Gy]',
           'D5%[Gy]',
           'D1%[Gy]',
           'D0.03cc[Gy]',
           'V25.946Gy[cc]']

results = [dvh_metrics.execute_query(metric) for metric in metrics]

print(results)
```

That's it! You can move on to the *user manual* to see what part of this library interests you.

3.1.5 DICOM-RT Parsing

PyPlanScoring inherited its dicom-parsing features from dicompyler-core dicomparser module. The class that encapsulates DICOM handling is `PyDicomParser`.

The method `PyDicomParser.GetStructures` returns a dictionary object that contains each structure indexed by roi number as key.

```
from pyplanscoring import PyDicomParser, PyStructure

rs_file = 'DICOM-RTStructure.dcm'
rs_dcm = PyDicomParser(filename=rs_file)

structures = rs_dcm.GetStructures()  # Dict like information of contours
```

PyPlanScoring provides the adapter class `PyStructure` to encapsulate methods that are not available on the container object dictionary that is returned by `PyDicomParser.GetStructures` method.

```
# encapsulate data on PyStructure object
structure = PyStructure(structures[1])
```

It is also possible to visualize DICOM-Structure contours if vispy package is installed. There is a helper method from `pyplanscoring.vis.contours3d`.

```
from pyplanscoring.vis.contours3d import plot_structure_contours
# encapsulate data on PyStructure object
structure = PyStructure(structures[1])
plot_structure_contours(structure.point_cloud, structure.name)
```

Example result of body contours extracted from DICOM-Structure file.

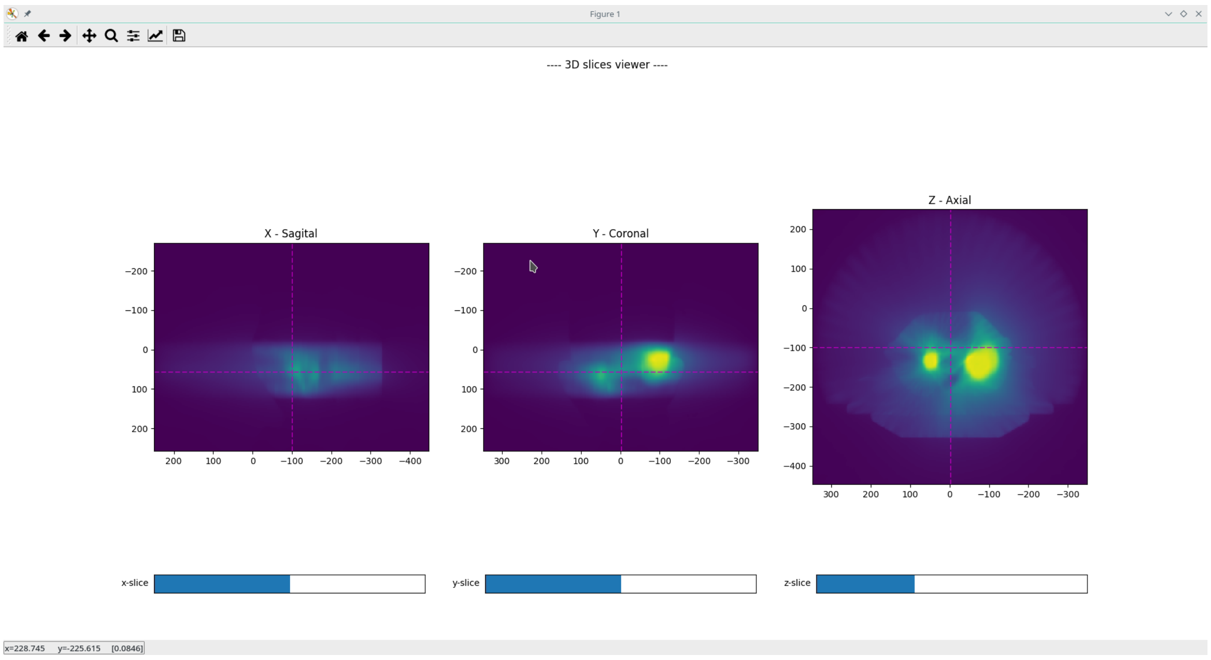
PyPlanScoring provides methods to handle DICOM-RTDOSE files. The class `pyplanscoring.core.types.Dose3D` adds a layer of abstraction on top of the dose-matrix provided by RTDOSE files. This class implements `scipy.interpolate.RegularGridInterpolator` to provide trilinear dose interpolation on regular grid coordinates.

There is a factory class `pyplanscoring.DoseAccumulation` to provide plan-sum capabilities, by using operator overloading on `pyplanscoring.core.types.Dose3D`

It is possible to visualize axial, coronal and sagittal slices from a Dose3D matrix.

```python
from pyplanscoring import PyDicomParser, DoseAccumulation
from pyplanscoring.vis.slice3 import DoseSlice3D

dose_files = ["Plan1_dose.dcm", "Boost.dcm"]

rd_dcm = [PyDicomParser(filename=dose_file) for dose_file in dose_files]
doses_obj = [d.get_dose_3d() for d in rd_dcm]

# using factory class to add 3d doses matrix
acc = DoseAccumulation(doses_obj)
dose_sum = acc.get_plan_sum()

# View the result
dose_view = DoseSlice3D(dose_sum)
dose_view.show()
```

Example result:



3.2 Contributing to PyPlanScoring

PyPlanScoring is a community-driven project on GitHub. You can find our repository on GitHub. Feel free to open issues for new features or bugs, or open a pull request to fix a bug or add a new feature.