SR-Tools: A Python Package for Space Resources

Scott Dorrington[1]

[1]Independent Postdoctoral Researcher, s.dorrington@unswalumni.com

Introduction

This poster presents an overview of the development of an open source Python software package aiming to provide a set of data management, analytics, modelling, and visualisation tools for research into space resource identification and utilization. The package currently consists of five core modules: **Datasets**, **Analytics**, **Plotting**, **Planetary Bodies**, and **Astrodynamics**.

Datasets Module

The **Datasets** module provides a set of classes that automate the downloading and loading of asteroid and planetary data. Datasets are loaded from various online sources into Pandas dataframes that are then formatted to provide a consistent namespace for the various data fields and designations. Datasets can be stored on the user's machine as either csv files or tables in a PostgreSQL database to allow for fast loading of entire datasets and querying of individual objects. Data aggregation methods are employed to combine all available data into new aggregated datasets collecting similar data types such as orbital, photometric, color, and physical parameters. Utility methods allow users to list available datasets (30 implemented so far) and search field variable descriptions. The code listing below demonstrates the loading of the Minor Planet Center Orbital Elements Database [1].

```
from sr_tools.Datasets import Datasets
# Load the Minor Planet Center Orbital Elements Database
df = Datasets.load_dataset('MPCORB_table',version='current')
print(df) # Print the dataframe
```

```
pdes designation          a         e          i ...    H     epoch
1       (1) Ceres   2.766089  0.078168   10.58790 ...  3.53  2459200.5
2       (2) Pallas  2.774382  0.229750   34.85446 ...  4.21  2459200.5
3       (3) Juno    2.668020  0.256988   12.99149 ...  5.26  2459200.5
4       (4) Vesta   2.362030  0.088425    7.14165 ...  3.29  2459200.5
5       (5) Astraea 2.573621  0.190788    5.36754 ...  6.99  2459200.5
...     ...               ...       ...        ... ...   ...        ...
[1046550 rows x 40 columns]
```

Analytics, Plotting & Astrodynamics Modules

The **Analytics** module provides tools to apply data analytics methods to the compiled datasets, including statistical analysis, clustering, and classification. The **Plotting** module provides methods to visualize the datasets and results from the analytics module. These include both static plots, as well as interactive web-based plots using the python Plotly and Dash packages. The **Astrodynamics** module provides a set of utility functions for working with planetary ephemeris data, vectorized orbital mechanics functions, and trajectory optimization methods including Porkchop plots.

Planetary Bodies Module

The **Planetary Bodies** module contains classes representing object models of planetary bodies including **Planets**, **Moons**, and **Asteroids**. Each class is parsed a name or designation that is used to query all available information from the internal PostgreSQL database as well as external publicly available databases. The data are organized into groupings such as orbital, photoemtric, spectral, and physical parameters that are appended as class attributes. Additional classes are used to model sub-components of the planetary body, including orbit/ephemeris, spectra, physical structure, and surface regions. These classes provide additional methods specific to each component. The code listing below demonstrates the creation of different planetary objects, and the use of a Dash apps to display their information along with interactive orbit and spectral plots (Fig 1).

```
from sr_tools.planetary_bodies import *
# Load basic data for planetary bodies
earth = Planet('Earth')    # Planet Earth
luna = Moon('Luna')        # The Moon
bennu = Asteroid('Bennu') # Asteroid 101955 Bennu
# Display Bennu's properties in a Dash app
bennu.run_app()
```

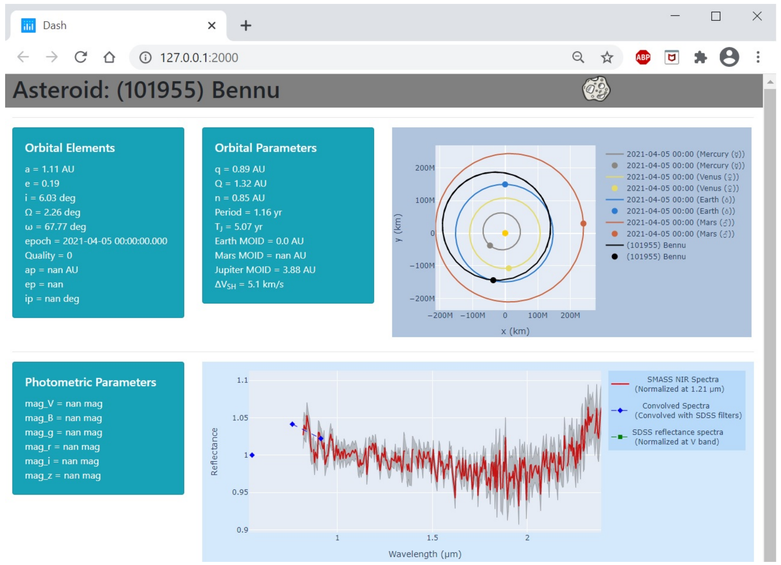

Fig. 1: Asteroid Dash app.

Physical Modelling

The bulk physical structure of the planetary bodies are modelled as one or more constant density polyhedra, generated from either a (.obj) shapefile or the dimensions of a tri-axial ellipsoid. The **ConstDensityBody** class contains methods to generate triangulated surface and tetrahedral volume meshes of the body, from which mass, volume, inertia tensor, and spherical harmonic gravity coefficients are computed. Additional methods allow for 3D rendering of the body using the VTK package, as well as multi-body dynamic simulations using Chrono::Engine [5]. The code listing below demonstrates adding a physical model to asteroid Bennu (shown in Fig. 2).

```
# Generate a body from shapefile with CI meteorite properties
CI_material = Material.from_meteorite(type='CI')  # properties
body = ConstDensityBody.from_shapefile('BENNUSHAPE-V1.0.obj',
                                       AngVel=np.array([0.,0.,0.5]),
                                       material=CI_material)
# Append the body to an asteroid model and display
bennu.SetBody(body)
bennu.Body.plot_vtk(field='U') # Color by gravitational potential
```

Planetary Regions & Terrain Modelling

The **PlanetaryRegion** class provides tools to organize various GIS data sources available for a defined region on a planetary body. The region's information and extent are defined in an xls configuration file, along with details of raster files and WMS layers to be included. The class provides methods to retrieve and save the WMS layers; reading, plotting and reprojection of rasters; and geoprocessing methods to compute slope, azimuth, and distance maps from DEM rasters. These can be used to create custom operational constraint maps. DEM files can further be used to generate a **PlanetaryTerrain** object, modelling the 3D surface of the planetary region (shown in the bottom of Fig. 2). This process is demonstrated in the code listing below.

```
# Generate a region model of the Lunar South Pole
region = PlanetaryRegion(../path/to/config.xls) # Instantiate
region.generate_wms_rasters() # Save requested wms layers to rasters
region.generate_dem_products(region.rasters[0]) # DEM-derived rasters
# Generate constraint map with max/min slope
region.generate_constraint_map(region.rasters[0],slope_const=[0.,20.])
# Add Terrain model from DEM using ENU coordinates
region.SetPlanetaryTerrain(region.rasters[0], mesh_frame='ENU')
region.terrain.plot_vtk() # Render terrain model (color by elevation)
```

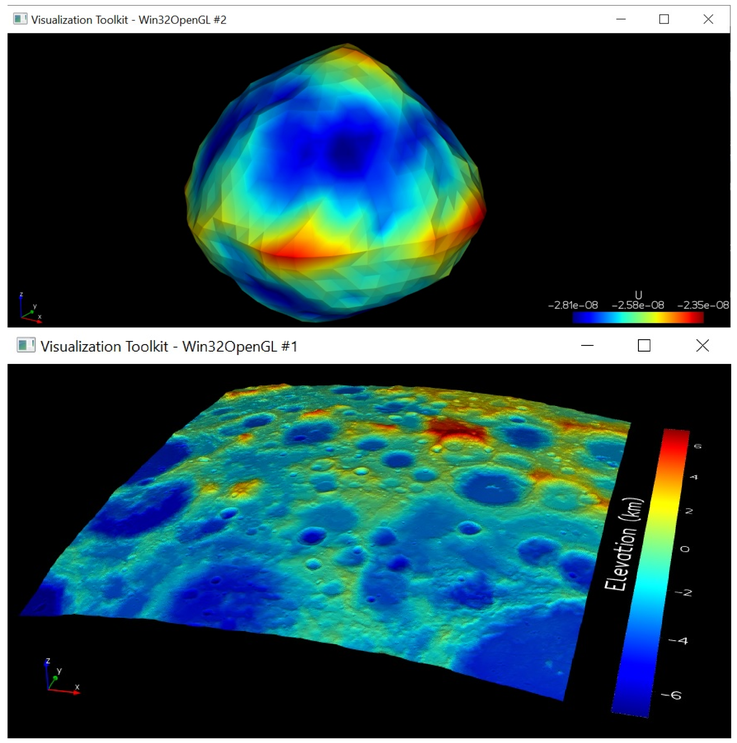

Fig. 2: 3D renderings of asteroid Bennu shape model (top) and a Lunar South Pole planetary terrain (bottom).

Conclusion

In its present format, the SR-Tools package provides a number of useful tools for accessing and organizing planetary data. Several classes have been implemented to model critical components of planetary bodies such as orbits, bulk structure, and surface regions. These classes may be used to construct a system of systems model representation of an entire planetary body, with future sub-classes modelling surface features such as craters and geological units. Current efforts are focusing on finalizing the software package for an initial public release (under the **SR-Tools Project** Github organization [2]). Future expansions of the package are expected to provide additional classes for modelling spacecraft and mining systems, simulations of resource extraction and processing methods, and tools for the operations planning and economic analysis of space resources missions.

Acknowledgements

Development of this software package was funded under the CSIRO Space Technology Future Sciences program, with collaboration from CSIRO, NEORA [3], HEO Robotics [4], and the University of Southern Queensland.

References

[1] Minor Planet Center. 2021. URL: https://minorplanetcenter.net/iau/MPCORB.html.

[2] S. Dorrington. *SR-Tools Project*. 2021. URL: https://github.com/SR-Tools-Project.

[3] NEORA. 2021. URL: http://www.neora.com.au/.

[4] HEO Robotics. 2021. URL: https://www.heo-robotics.com/.

[5] A Tasora. *Chrono:: Engine, an open source physics–based dynamics simulation engine*. 2006.