MOCLibRust: a common library for MOCPy, MOCCli and MOCWasm

F.-X. Pineau, M. Baumann, M. Allen, T. Boch, P. Fernique, G. Greco and A. Nebot

francois-xavier.pineau@astro.unistra.fr

Multi-Order Coverage map

Multi-Order Coverage map (MOC) is an IVOA standard and a powerful tool to create and manipulate discretized space, time and space-time (ST) coverages. For example, one can retrieve the pre-built ST-MOCs of XMM and Chandra and easily find the sky areas observed at the same time by both instruments.

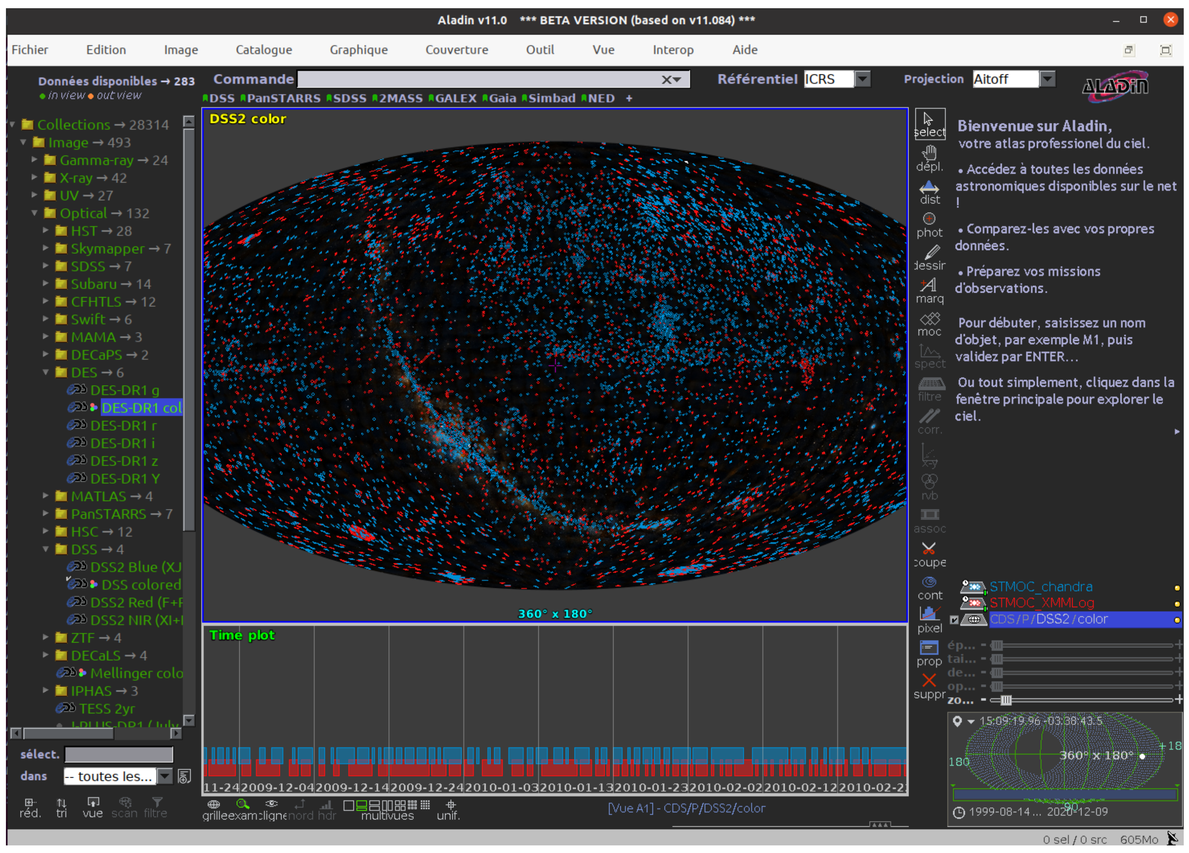

Figure 1: ST-MOCs of Chandra and XMM logs in Aladin. Aladin uses a Java MOC library.

The first version of the MOC standard deals exclusively with spatial coverages. The vesion 2.0 introduces Time MOCs and Space-Time MOCs. The last version of MOC 2.0 has been implemented both in a Java library (by Pierre Fernique) and in a Rust library currently used by MOCPy, MOCCli and MOCWasm. Check the MOCLibRust repo at https://github.com/cds-astro/cds-moc-rust.

MOCPy to manipulate MOCs from Python

MOCPy is a Python library allowing easy creation and manipulation of MOCs. It is available on both PyPI and Conda. Check the MOCPy repository at https://github.com/cds-astro/mocpy.

```python
from mocpy import MOC, World2ScreenMPL
from astropy.coordinates import Angle, SkyCoord
import astropy.units as u
# Load a MOC
filename = './../../resources/P-SDSS9-r.fits'
moc = MOC.from_fits(filename)
# Plot the MOC using matplotlib
import matplotlib.pyplot as plt
fig = plt.figure(111, figsize=(15, 10))
# Define a astropy WCS easily
with World2ScreenMPL(fig,
        fov=200 * u.deg,
        center=SkyCoord(0, 20, unit='deg', frame='icrs'),
        coordsys="icrs",
        rotation=Angle(0, u.degree),
        projection="AIT") as wcs:
    ax = fig.add_subplot(1, 1, 1, projection=wcs)
    # Call fill with a matplotlib axe and the `~astropy.wcs.WCS` wcs object.
    moc.fill(ax=ax, wcs=wcs, alpha=0.5, fill=True, color="green")
    moc.border(ax=ax, wcs=wcs, alpha=0.5, color="black")
plt.xlabel('ra')
plt.ylabel('dec')
plt.title('Coverage of P-SDSS9-r')
plt.grid(color="black", linestyle="dotted")
plt.show()
```

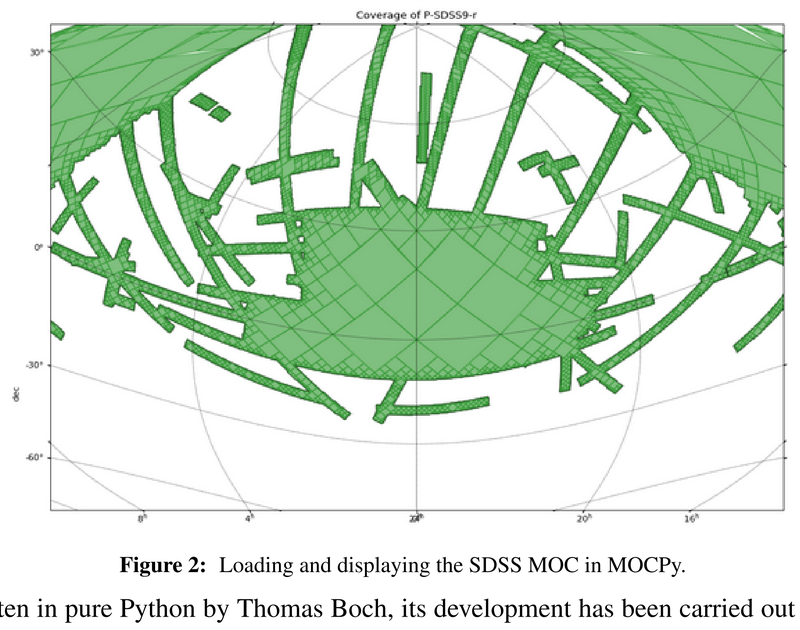

Figure 2: Loading and displaying the SDSS MOC in MOCPy.

Originally written in pure Python by Thomas Boch, its development has been carried out by Matthieu Baumann who also ported large parts in Rust to improve performances, and implemented an early version of the MOC 2.0 standard. *MOCLibRust* has been built by extracting and enlarging the MOCPy Rust core.

MOCCli to manipulate MOCs from the command line

MOCCli is a tool to load, create, manipulate and save MOCs from the command line. It consits in a single binary file pre-compiled for various architectures: MacOS; Windows; both 32 and 64 bits Linux. A *.deb* file is also available for Debian and derivatives such as Ubuntu.

```
pineau@cds-dev-fxp:~/tmp$ moc from cone --help
moc-from-cone 0.1.0
Create a Spatial MOC from the given cone

USAGE:
    moc from cone <depth> <lon-deg> <lat-deg> <r-deg> <SUBCOMMAND>

FLAGS:
    -h, --help       Prints help information
    -V, --version    Prints version information

ARGS:
    <depth>      Depth of the created MOC, in `[0, 29]`
    <lon-deg>    Longitude of the cone center (in degrees)
    <lat-deg>    Latitude of the cone center (in degrees)
    <r-deg>      Radius of the cone (in degrees)

SUBCOMMANDS:
    ascii      Output an ASCII MOC (VO compatible)
    fits       Output a FITS MOC (VO compatible)
    help       Prints this message or the help of the given subcommand(s)
    json       Output a JSON MOC (Aladin compatible)
    stream     Output a streamed MOC
pineau@cds-dev-fxp:~/tmp$ moc from cone 11 083.63308 +22.01450 0.25 ascii --fold 80
8/386969
9/1547853-1547855 1547864 1547866 1547888-1547890
10/6191405 6191407 6191409-6191411 6191462 6191468 6191470-6191471 6191493
 6191495 6191540-6191541 6191564 6191566 6191584
11/24765562-24765563 24765566-24765567 24765597 24765599 24765627
 24765634-24765635 24765738 24765842 24765854 24765876 24765878-24765879
 24765969 24765971 24766004-24766005 24766007 24766013 24766148-24766149
 24766260-24766262 24766340-24766341
pineau@cds-dev-fxp:~/tmp$
```

Figure 3: Create and display the ASCII serialization of the MOC of a given cone.

Check MOCCli releases at https://github.com/cds-astro/cds-moc-rust/releases and the source code at https://github.com/cds-astro/cds-moc-rust/tree/main/crates/cli.

MOCWasm to manipulate MOCs from Web Browsers

MOCWasm is a WebAssembly library to load, create, manipulate and save MOCs from JavaScript. It can be used direclty from the console of your favorite Web browser, or could be included in an user interface such as Aladin Lite. Check the project at https://github.com/cds-astro/cds-moc-rust/tree/main/crates/wasm and the demo web page at http://cdsxmatch.u-strasbg.fr/lab/moc/.

```
// Load 2MASS and SDSS DR12 MOCs from CDS
await moc.fromFitsUrl('2mass', 'http://alasky.u-strasbg.fr/footprints/tables/vizier/II_246_out/MOC');
await moc.fromFitsUrl('sdss12', 'http://alasky.u-strasbg.fr/footprints/tables/vizier/V_147_sdss12/MOC');

// List MOCs loaded in the page
console.log(moc.list());

// Init a timer
console.time('timer');
// Performs MOC intersection
moc.and('2mass', 'sdss12', '2mass_inter_sdss12');
// Log time
console.timeLog('timer', 'Intersection');
// Performs MOC union
moc.or('2mass', 'sdss12', '2mass_union_sdss12');
// Log time
console.timeLog('timer', 'Union');
// Degrade to order 2 the result of the intersection
moc.degrade('2mass_inter_sdss12', 2, '2mass_inter_sdss12_d2')
// Remove timer
console.timeEnd('timer');

// List MOCs loaded in the page
console.log(moc.list());

// Print the ASCII and JSON serializations of '2mass_inter_sdss12_d2'
console.log(moc.toAscii('2mass_inter_sdss12_d2'));
console.log(moc.toJson('2mass_inter_sdss12_d2'));

// Save the result of the intersection in a FITS file
moc.toFitsFile('2mass_inter_sdss12');
```

```
// Load 2MASS and SDSS DR12 MOCs from CDS
await moc.fromFitsUrl('2mass', 'http://alasky.u-strasbg.fr
await moc.fromFitsUrl('sdss12', 'http://alasky.u-strasbg.f

// List MOCs loaded in the page…
Array [ "2mass", "sdss12" ]
timer: 0ms Intersection
timer: 0ms Union
timer: 1ms - timer ended
Array(5) [ "2mass", "2mass_inter_sdss12_d2", "2mass_inter_
0/1-4 7 1/0-2 21-23 25-27 2/12 14-15 81 83 97-99 139 141-143
{
  "0": [1, 2, 3, 4, 7],
  "1": [0, 1, 2, 21, 22, 23, 25, 26, 27],
  "2": [12, 14, 15, 81, 83, 97, 98, 99, 139, 141, 142, 143,
}
undefined
```

Opening 2mass_inter_sdss12.fits

You have chosen to open:

2mass_inter_sdss12.fits

which is: document FITS (84.2 KB)
from: blob:

What should Firefox do with this file?

Open with ImageMagick (color depth=q16) (default)

Save File

Do this automatically for files like this from now on.

Cancel OK

Figure 4: Top: Javascript calls to the MOCWasm library from the Firefox console. Bottom: Result of the execution of the calls.

Acknowledgements

This work has been partly supported by the ESCAPE project.
ESCAPE - The European Science Cluster of Astronomy & Particle Physics ESFRI Research Infrastructures has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 824064.

François-Xavier Pineau (francois-xavier.pineau@astro.unistra.fr)
Université de Strasbourg, CNRS, Observatoire astronomique de Strasbourg, UMR 7550, F–67000 Strasbourg, France

ADASS XXXI, Astronomical Data Analysis Software & Systems
24 – 28 October 2021, Cape Town, South Africa and Online