MOCLibRust, a common library for MOCPy, MOCCli and MOCWasm

Francois-Xavier Pineau, Matthieu Baumann, Mark G. Allen, Thomas Boch, Fernique Pierre, Giiuseppe Greco, Ada Nebot

MOCLibRust, a common library for MOCPy, MOCCli and MOCWasm

F.-X. Pineau,[1] M. Baumann,[1], M. Allen,[1], T. Boch,[1], P. Fernique,[1], G. Greco,[2] and A. Nebot,[1]

[1]*Université de Strasbourg, CNRS, Observatoire astronomique de Strasbourg, UMR 7550, F-67000, Strasbourg, France;*
francois-xavier.pineau@astro.unistra.fr

[2]*INFN - Sezione di Perugia, Italy*

Abstract. Multi-Order Coverage map (MOC) is an International Virtual Observatory Alliance (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 – and quickly – find the sky areas observed at the same time by both instruments.

So far, two tools have implemented time and space-time MOCs: the Java Library used in Aladin, and MOCPy. Originally written in pure Python, a part of MOCPy has been rewritten as a wrapper calling Rust code. This effort has been pursued, resulting in an independent Rust library: MOCLibRust. The main motivations were to improve performances and to be able to reuse the same codebase in different tools. In addition to MOCPy, MOCLibRust is now used in MOCCli, a standalone command line program, and in MOCWasm, a WebAssembly library to manipulate MOCs from JavaScript (and thus from tools like Aladin Lite). In the future, we may also consider the development of C and/or PostgreSQL wrappers.

1. Multi-Order Coverage map

The MOC IVOA standard (Fernique et al. 2019) originally describes a method to specify discretized regions on the unit sphere. It relies on the hierarchical properties of the HEALPix tessellation (Górski et al. 2005).
While the first version of the MOC standard deals exclusively with spatial coverages, the version 2.0 introduces Time MOCs and Space-Time MOCs. The last MOC 2.0 version has been implemented independently in both a Java library[1] used in Aladin, and in a Rust library, *MOCLibRust*. *MOCLibRust* is a currently used by *MOCPy*, *MOCCli* and *MOCWasm*. We briefly introduce those three tools in the following sections.
The current status of the MOC standard is available on the IVOA web site[2], while the *MOCLibRust* status can be checked from its github repository [3].

[1]https://wiki.ivoa.net/twiki/bin/view/IVOA/MocInfo

[2]https://www.ivoa.net/documents/MOC/20210324/index.html

[3]https://github.com/cds-astro/cds-moc-rust

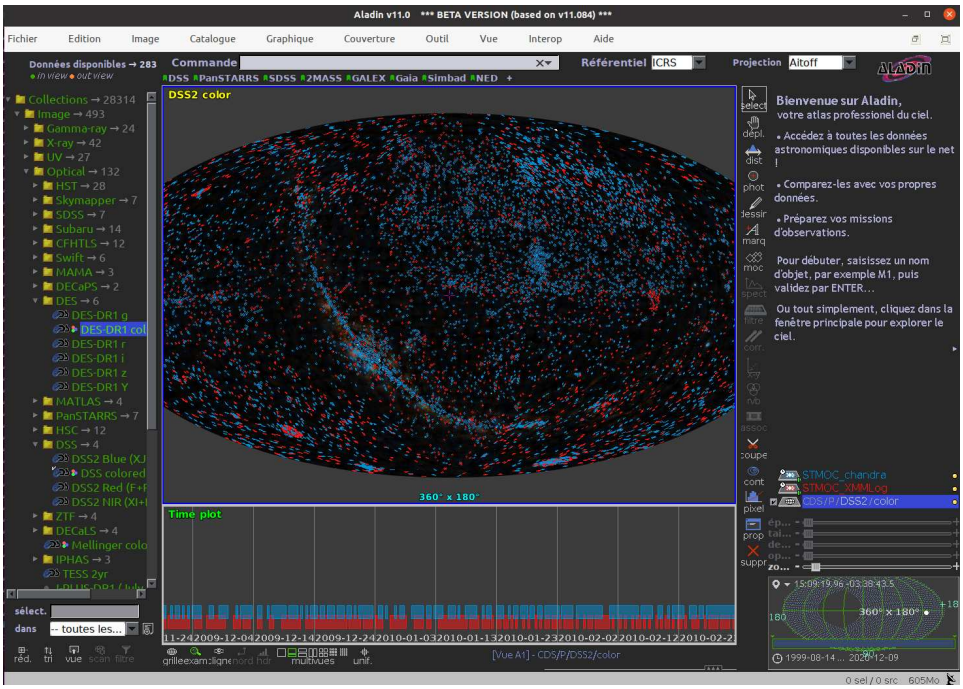

Figure 1. ST-MOCs of Chandra and XMM logs in Aladin.

2. MOCLibRust and associated tools

2.1. MOCLibRust

MOCLibRust is a pure Rust implementation of core MOC algorithms like the computation of the intersection or union of two MOCs, Time MOCs, or ST-MOCs.
It relies on the CDS HEALPix Rust library[4] (Pineau & Fernique 2019) to create MOCs from spherical shapes like cones or polygons.

The rust MOC library implements lazy operations resorting on iterators, is generic enough to support additional quantities (i.e. quantities other than space coordinates or timestamps) and supports several indexation data type such as integer or long integer. It also supports the creation of MOCs from Multi-Resolution HEALPix Maps (Martinez-Castellanos et al. 2021) and innovative functionalities such as splitting a MOC into a list of disjoint MOCs.

Rust is a compiled language similar in syntax and performances to C++ but benefiting from memory safety guarantees. Once compiled, a Rust code is similar to a C static library. It can thus interact with other language like a C code, and can be directly compiled into WebAssembly (wasm).

[4] `https://github.com/cds-astro/cds-healpix-rust`

2.2. MOCPy

MOCPy[5] is an open source Python library allowing easy creation and manipulation of MOCs. It is available on both *PyPI*[6] package manager and the *anaconda* platform[7].

Originally written in pure Python by Thomas Boch (Boch 2019), 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 (Baumann et al. 2020).

MOCLibRust has been built by extracting and enlarging the *MOCPy* Rust core to support the last version of the MOC 2.0 standard and to natively serialize and deserialize FITS, JSON and ASCII MOC formats.

2.3. MOCCli

MOCCli is an open source tool made to load, create, manipulate and save MOCs from the command line, without requiring a Python environment. It consists in a single binary file pre-compiled for various architectures: MacOS; Windows; both 32 and 64 bits Linux. A *.deb* file is also available to install the binary file and associated documentation on Debian systems and derivatives such as Ubuntu.

MOCCli source code and releases – including *.deb* and binary files – are available on github [8] [9].

2.4. MOCWasm

MOCWasm is a WebAssembly library made to load, create, manipulate and save MOCs from *JavaScript*. It can be used directly from a Web browser's console, or could be included in user interfaces such as Aladin Lite. The project is open source and available on github [10]. To use *MOCWasm* in a web page, one have to download a *.js* and a *.wasm* file and to call the *.js* file at page loading. See the project *README* file for more information.

Acknowledgments. 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.

References

Baumann, M., Boch, T., Fernique, P., Nebot, A., & Pineau, F. X. 2020, in Astronomical Data Analysis Software and Systems XXIX, edited by R. Pizzo, E. R. Deul, J. D. Mol, J. de Plaa, & H. Verkouter, vol. 527 of Astronomical Society of the Pacific Conference Series, 693

---

[5] `https://github.com/cds-astro/mocpy`

[6] `https://pypi.org/project/MOCPy`

[7] `https://anaconda.org/conda-forge/mocpy`

[8] `https://github.com/cds-astro/cds-moc-rust/releases`

[9] `https://github.com/cds-astro/cds-moc-rust/tree/main/crates/cli`

[10] `https://github.com/cds-astro/cds-moc-rust/tree/main/crates/wasm`

```
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 2. Create and display the ASCII serialization of the MOC of a given cone with MOCCli.

Boch, T. 2019, in Astronomical Data Analysis Software and Systems XXVI, edited by M. Molinaro, K. Shortridge, & F. Pasian, vol. 521 of Astronomical Society of the Pacific Conference Series, 487

Fernique, P., Boch, T., Donaldson, T., Durand, D., O'Mullane, W., Reinecke, M., & Taylor, M. 2019, MOC - HEALPix Multi-Order Coverage map Version 1.1, IVOA Recommendation 07 October 2019

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., & Bartelmann, M. 2005, ApJ, 622, 759. astro-ph/0409513

Martinez-Castellanos, I., Singer, L. P., Burns, E., Tak, D., Joens, A., Racusin, J. L., & Perkins, J. S. 2021, arXiv e-prints, arXiv:2111.11240. 2111.11240

Pineau, F.-X., & Fernique, P. 2019, in Astronomical Data Analysis Software and Systems XXVII, edited by P. J. Teuben, M. W. Pound, B. A. Thomas, & E. M. Warner, vol. 523 of Astronomical Society of the Pacific Conference Series, 609