Mastering Radio Astronomy Simulations with Karabo

Andreas Wassmer
andreas.wassmer@fhnw.ch

University of Applied Sciences and Art Northwestern Switzerland

September 3, 2024

Outline

About me

- Andreas Wassmer
- Physicist, worked in software industry
- Joined FHNW Karabo team in January 2024
- Focus on user experience
- Find me on LinkedIn and Xing

Where to get the code?

Link to the workshop's Jupyter notebooks and server

`https://renkulab.io/projects/menkalinan56/swissskadays-karabo-workshop`

There is also the code and documentation of this workshop.

Documentation for Karabo is available on:

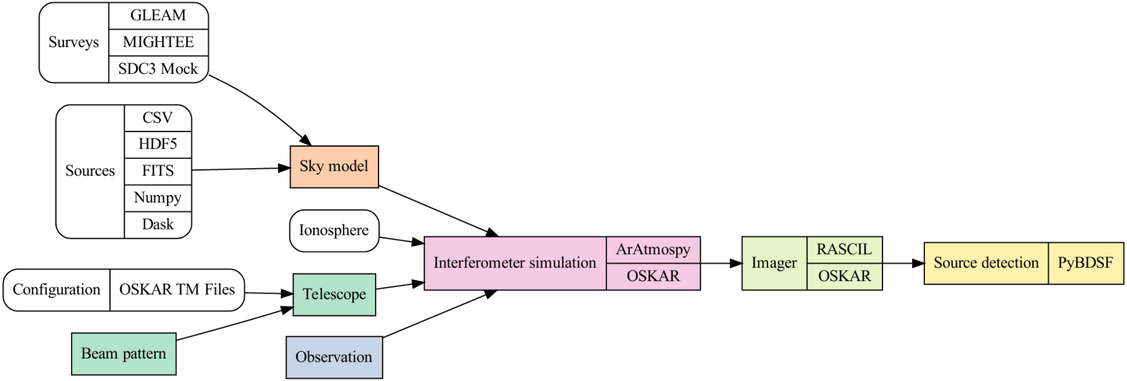
`https://i4ds.github.io/Karabo-Pipeline/`

Goals of Karabo

- Make it easy to set up a data processing pipeline for radio astronomy
- Easy way to compare established and new algorithms
- Helps to develop and test new ideas quickly

Current version is 0.27.0. There are breaking changes from 0.24.0!

Karabo Workflow



How to run Karabo?

1. Install Karabo in conda environment
2. Run it within a container (Docker, Singularity)
3. Create an account on renkulab.io, login and search for Karabo. This is great if you want to explore the possibilities of Karabo. We use this for this workshop.

Details on how to install can be found in the documentation:
`https://i4ds.github.io/Karabo-Pipeline/installation_user.html`

How to login to Renkulab?

No account needed, but you will not be able to save your work:

1. Go to `https://renkulab.io/`, click on "Sessions", then "Search" and search for "SwissSKADays"
2. Take the one which is authored by me (Andreas Wassmer, aka menkalinan56) and hit "Start"

If you want to save your work, you must login to Renkulab. You can do this either with:

1. a GitHub login
2. a Switch edu-ID or
3. an ORCID ID.

Then search for "SwissSKADays" and fork the project. Now you can work with your own copy.

Sky Models

Sky models available in Karabo

- GLEAM Survey
- MALS Survey V3
- MIGHTEE Survey
- HI Sources (small catalog, simulated sky)

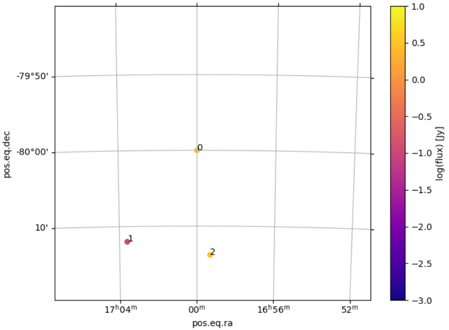
Examples are in notebook `2_skymodel/skymodels.ipynb`.

You can always setup your own sky

```
sky = SkyModel()
sky_data = np.array(
  [
    # Required columns:
    # =================
    # RA(deg), Dec(deg), I(Jy)
    #
    # Optional columns:
    # =================
    # Q(Jy), U(Jy), V(Jy), freq0(Hz), spectral index, rotation
          measure,
    #   FWHM major (arcsec), FWHM minor (arcsec), position angle
          (deg)
      [255, -80.0, 3, 0, 0, 0, 100.0e6, -0.7, 0.0, 0, 0, 0],
      [255.9, -80.2, 3, 2, 2, 0, 100.0e6, -0.7, 0.0, 600, 50, 45],
  ])

  sky.add_point_sources(sky_data)

  # A source with only Stokes I (other attributes take default
        values)
  sky.add_point_sources(np.array([[254.83,-80.23, 3]]))
```



Using Telescopes

Karabo offers predefinded telescope settings

- with OSKAR backend: MeerKAT, ASKAP, SKA1LOW, SKA1MID, ...
- with RASCIL backend: LOW, ASKAP, LOFAR, MID, ...

Definitions of OSKAR telescopes are on Github
https://github.com/i4Ds/Karabo-Pipeline/tree/main/karabo/data

Defining a Telescope

You can define an instrument setup with OSKAR tm folder

```
telescope.tm
├── layout.txt
└── position.txt
    ├── station_1
    │   └── layout.txt
    ├── station_2
    │   └── layout.txt
    └── station_nnn
        └── layout.txt
```

The layout definiton is part of the OSKAR package. If you want to learn more visit https://ska-telescope.gitlab.io/sim/oskar/telescope_model/telescope_model.html#layout-files

Hands on

See notebook `notebooks/1_telescope/simple-telescope.ipynb`

```
from karabo.simulation.telescope import Telescope
from karabo.simulator_backend import SimulatorBackend
...
BACKEND = SimulatorBackend.OSKAR
telescope = Telescope.constructor("MeerKAT", backend=BACKEND)
```

Setup an Observation

Now we have 2 out of 3: a sky model and the telescope. Next is to set up an observation. Mandatory parameters are:

- phase center (RA and Dec. in degrees)
- start date and time
- start frequency (in Hertz)
- frequency increment (in Hertz)
- observation length (in secs)

Define Simulation

Now we are ready to set up and run the simulation. The result is a file with visibilities.

```
visibility = simulation.run_simulation(
    telescope,
    sky_model,
    observation,
    backend=BACKEND,
)
```

For `RASCIL` only: You can supply an arbitrary primary beam defined in a FITS file if you use parameter `primary_beam="beam.fits"`

General Support for Primary Beam

You can use a limited set of primary beams when using the OSKAR backend. Choices are isotropic beam, Gaussian beam, aperture array or VLA (PBCOR). You need to make the choice in the constructor of `InterferometerSimulation`

```
simulation = InterferometerSimulation(
    station_type="Gaussian beam",
    gauss_beam_fwhm_deg=1.1832497493784,
    gauss_ref_freq_hz=1.34e9,
)
```

Next step is to calculate the dirty image.

Getting the Dirty Image

Karabo offers 3 algorithms

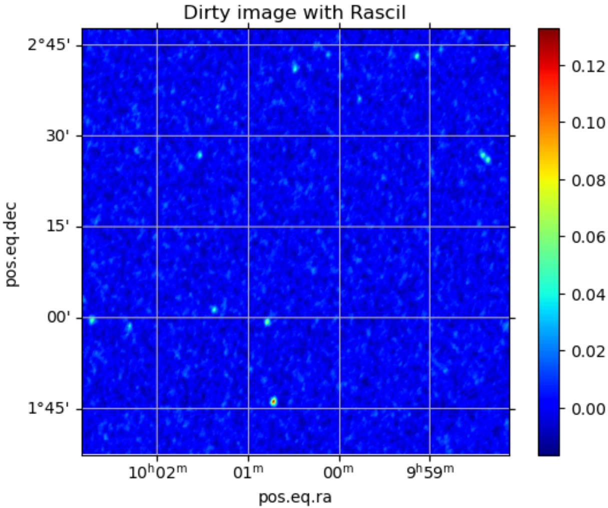
- `OSKARDirtyImager`
- `RascilDirtyImager`
- `WscleanDirtyImager`

See example in `3_imaging/dirty_image.ipynb`

Dirty Image with Rascil Backend

```
from karabo.imaging.imager_rascil import RascilDirtyImager,
    RascilDirtyImagerConfig
...
rascil_dirty_imager = RascilDirtyImager(
    RascilDirtyImagerConfig(
        imaging_npixel=imaging_npixel, # e.g. 4096
        imaging_cellsize=imaging_cellsize, # e.g. FOV / 4096 = 5e-6
        combine_across_frequencies=True,
    )
)
dirty_image = rascil_dirty_imager.create_dirty_image(visibility)
dirty_image.write_to_file("output/rascil_dirty_image.fits")
```

Result


Dirty image with Rascil
2°45'
30'
15'
00'
1°45'
pos.eq.dec
10h02m
01m
00m
9h59m
pos.eq.ra
0.12
0.10
0.08
0.06
0.04
0.02
0.00

Getting the clean image

You may require a clean image. You can choose from 2 algorithms:

- `RASCIL`
- WSClean (currently only works for OSKAR visibilities)

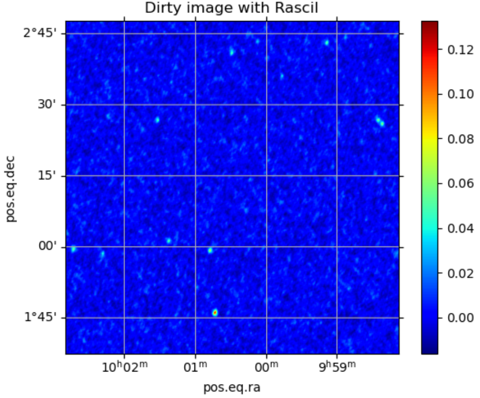
See at the end of example in `3_imaging/dirty_image.ipynb` for how to do it.

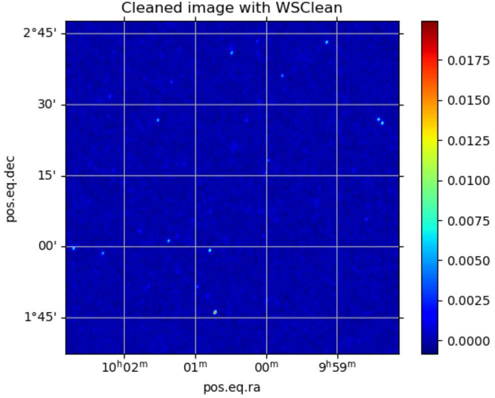
How to

```
...
cleaned_image = WscleanImageCleaner(
    WscleanImageCleanerConfig(
        imaging_npixel=imaging_npixel,
        imaging_cellsize=imaging_cellsize,
    )
).create_cleaned_image(
    ms_file_path=visibility.ms_file_path,
)

cleaned_image.write_to_file("output/cleaned_image.fits",
    overwrite=True)
```

Result

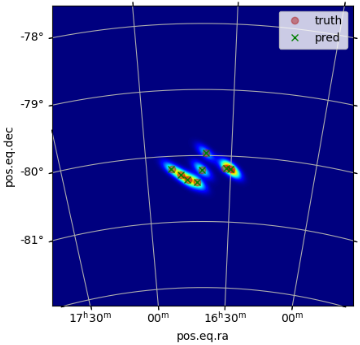

Dirty image with Rascil
2°45'
30'
15'
00'
1°45'
pos.eq.dec
10h02m
01m
00m
9h59m
pos.eq.ra
0.12
0.10
0.08
0.06
0.04
0.02
0.00


Cleaned image with WSClean
2°45'
30'
15'
00'
1°45'
pos.eq.dec
10h02m
01m
00m
9h59m
pos.eq.ra
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000

But there is more

Karabo comes with **automatic source detection**.

```
detection_result =
    PyBDSFSourceDetectionResult.detect_sources_in_image(
  image=restored_wsclean,
  thresh_isl=15,
  thresh_pix=20,
)
```



Karabo also offers validation tools such as confusion matrix and assignment of ground truth and predictions. See example in `4_pipelines/source_detection.ipynb`.

Features to come

Developement of Karabo continues. Currently, we are working on

- supporting IVOA ObsCore Metadata
- consistent handling of measurement sets throughout the entire workflow
- switching to numpy version 2.0

Where to go from here?

Your will keep your account on Renkulab, if you don't delete it. So you'll have a playground for Karabo. Why not:

- play around with the `telescope` notebook and have a look at the instruments available?
- generate a different sky model?
- adapt and run the `simulation_workflow.ipynb` workflow?
- run a HI sky simulation?
- look at more examples on `https://i4ds.github.io/Karabo-Pipeline/examples/examples.html`?

Questions, Remarks and Features

Code and documentation

Check out the project repository

`https://github.com/i4Ds/Karabo-Pipeline/tree/main`

Do you have any questions or remarks?
Not sure how to use Karabo for your use case?
Would you like to see a feature in Karabo?

I am happy to help: `andreas.wassmer@fhnw.ch`

Thank you very much!