Karabo Report

ARLT Patrick

Introduction:
I am currently a student in second year of Master is computer sciences at the University of Geneva. As it is my last year here, I had to choose a Master Thesis subject which is about astronomical radio sources detection and in collaboration with Olga Taran.

In a first time, I had to read about astronomical radio data and different source detection method. Then, the goal was to discover Karabo, install it and start playing with it.

The installation:
I tried installing Karabo but had a lot of struggle doing so. To clarify the situation, I am currently working on MacOS and apparently after opening an issue on Karabo's github (https://github.com/i4Ds/Karabo-Pipeline/issues/167), they told me there was an issue the package Rascil which was the culprint. They then suggest me to download the source code and do some modification (such as commenting lines importing Rascil and so). With this temporary solution, I still couldn't use Rascil and the simulation tools. But in our case, we need to use Karabo only for the detection purpose and so the simulation tools were not useful for us.

I then tried running the Jupiter Notebook example but had new issues with OSKAR this time and figured out that the code of the example was outdated.
In the meantime, I also figured out that the installation of PyBDSF didn't went as planned and some files were missing. That's where I tried reinstalling it but had new struggles doing so (refer to this issue https://github.com/lofar-astron/PyBDSF/issues/183 to get more details). Long story short, I wasn't able to install PyBDSF on my conda environment for whatever reason (maybe there was some conflicts or so) but succeed installing it on my base environment and on a fresh new environment.

At this point, I could start playing with Karabo for source detection purpose.

Experiments:
The first thing I did was to see hot to plot and use fits files (using simply Matplotlib or using Karabo). Then Miss Taran gave me a data set to work on. My job was to process some source detection on this set of fits file and compare the sources found using Karabo with the ground truth coordinates. At first, I explored how to perform the detection and the parameters I could change and use to tune the detection process.

To perform the detection, I used the function: ' SourceDetectionResult.detect_sources_in_image()' which takes only 3 parameters:

- The Image on which we want to perform the source detection;
- The quiet parameter (that only impacts how much prints are displayed;
- The beam parameter, which I explored but didn't seemed to bring any amelioration to the detection when specifying the exact beam of the image.

I was quite surprised to see that it was the only parameters we could play with, knowing the Karabo is using PyBDSF and that PyBDSF as a lot of parameters we can use to tuned the detection.

Once the detection has been performed, we can extract the coordinates of the sources detected using the function 'get_pixel_position_of_sources()' and one astonishing thing is that this function

name literally says that we can extract pixels values (which should be represented as Integers) but are in reality real numbers.

Also the ground truth values of the position of the source were given in real units (Ra & Dec) and not in pixels. That's why I tried searching for a Karabo function that could give us the coordinates of the source in the real units instead of pixels, but unfortunately, I didn't find such a function. I then had to create a function that converts Ra & Dec coordinate to Pixels coordinates and also a function the perform this conversion the other way around.

Note that I did this sources detections on two different dataset (of size 916 fits files):

- On clean noiseless dataset
- On a clean noisy datasat

Results:

Anyways, I then compare the pixels coordinates of the detected sources with the ground truth values. If the distance between the detected source and the real sources was smaller than a radius of 10 pixels, I consider this as a correct prediction. If Karabo detects sources that isn't existing in real data, this was count as False Positive (FP) and if Karabo didn't detect a source that exists in reality, I considered this as False Negatives (FN).

I then was able to compute the purity and the completeness of both source detection (on the noisy and the noiseless dataset) and obtained the following results:

	Purity	Completeness
Clean noiseless	0.3433	0.9806
Clean noisy	0.9772	0.1661

	Number of TP	Number of FP	Number of FN
Clean noiseless	2781	5320	55
Clean noisy	471	11	2365

We can conclude easily that with the basic parameters Karabo uses from PyBDSF, It tend to detect much more source that there really is in the clean noiseless dataset, and that it has some struggles detecting sources in the noisy dataset (as there is much more false negative than true positive).

Also note that I found out that Karabo wasn't handling correctly the case where it can't detect any source. Indeed when Karabo perform the detection, It then extract the pixel coordinates using indexes (If I did understand the source code right). But this extraction expects to obtain an array that is 2-dimesionnal (one list of 'x' coordinates and one of 'y' coordinates). But when there is no sources detected, we obtain simply an empty array (represented like so '[ ]') and so it has only 1 dimension. This causes an 'IndexError' error when running the function 'SourceDetectionResult.detect_sources_in_image()' and that no source is detected.

Conclusion:

From my experience, Karabo seems quite easy to use (even If the installation part still has some issues) but in my opinion, it would be really great to have a well written documentation somewhere so we don't have to search in the source code (which is what I did most of the time as the examples were outdated and that there isn't really a real documentation yet).

Then, a huge negative point for Karabo is that we can't tune the source detection knowing that PyBDSF has a lot of parameter with which we can play with to get better results.
Also, the fact that we can extract the source coordinates only in pixels value and not in Ra & Dec units is unfortunate, even if the returned values are easy to use then. Another remark on this is that the function ' get_pixel_position_of_sources()' returns an array of shape (2, number of sources detected) which would be more intuitive the other way around: (number of sources detected, 2).

A small improvement as well that shouldn't be too hard do would be to handle the fact when the detection doesn't find any sources.
Anyways, Karabo has some potential in my opinion but still need some work and improvement before using it if your goal is to have a good source detection, even if It is quite easy to understand how it works and easy to use.

Note that I didn't make any remark on the simulation tools due to the fact that I wasn't able to use and play with it.

Thank you for taking the time of reading this,
You can contact me at 'patrickarlt@hotmail.fr' for any additional information.

ARLT Patrick
Master student at UNIGE
29.10.2022